



TECH TRENDS

2025



OCTO PULSE

OCTO donne le pouls de la **TECH.**

Épisode #1 — Modern
Software Engineering
& Architecture

OCTO TECHNOLOGY | PART OF ACCENTURE

Generated by Playground AI — Copyright @OCTO Technology



| AVANT-PROPOS |

Cet ouvrage a pour vocation de partager notre point de vue, synthétique mais surtout pas superficiel, sur les **Tech Trends 2025** du **Software Engineering**.

Il a pour ambition d'être **LA lecture de l'été** que vous emporterez dans votre **smartphone** pour décrypter les **tendances TECH** et préparer au mieux votre **roadmap 2025**. Notez qu'une version Web accessible de ce document sera publiée prochainement.

Il incarne le résultat de nos expériences terrain, d'échanges constants et de désaccords sains : un contenu **agnostique, indépendant et objectif** sans fausses promesses et sans langue de bois... même (et surtout) quand ça pique.

« La mode se démode, le style jamais. »
— Coco Chanel



Une année 2025

qui s'annonce sous le signe
de la **complexité**, avec :

La démocratisation
de la **GenAI**

Un **contexte socio-
économique**
sous **tensions**

De forts enjeux sur
le **Sustainable**
et la **régulation IT**

Une **archipélisation**
des **lieux de travail**
post-Covid



| SOMMAIRE |

01

IA GÉNÉRATIVE

- i. Développement augmenté par l'IA
- ii. Retrofit de la GenAI dans les apps

02

ARCHITECTURE

- i. Architectures composables
- ii. Architectures continues

03

EFFICIENCE OPÉRATIONNELLE & SUSTAINABLE

- i. erooM
- ii. Brownfield
- iii. Accessibilité numérique
- iv. Développement Hybride

04

ORGANISATION

- i. Software engineering distribué
- ii. T-shaped skills
- iii. Career path Tech

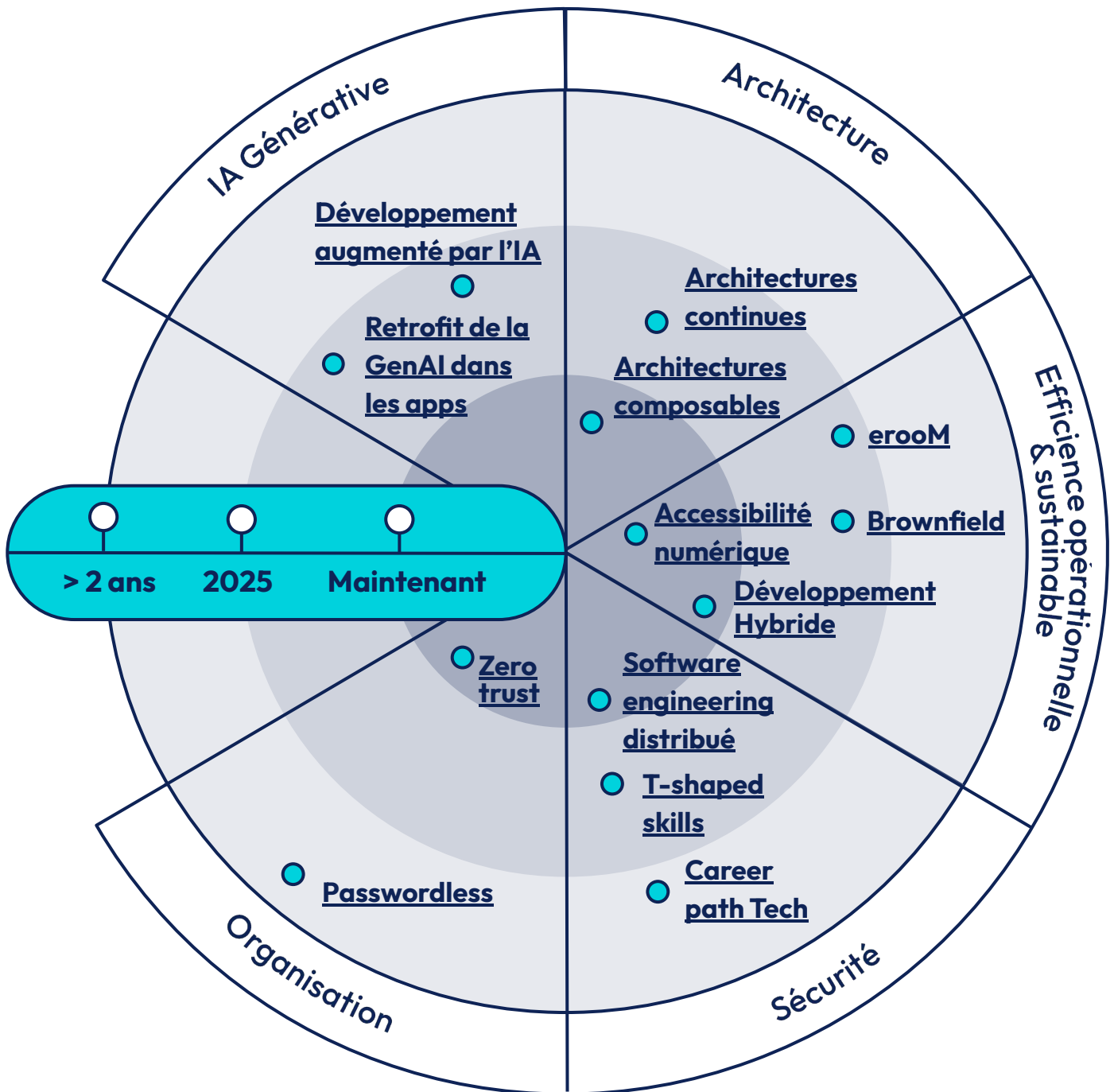
05

SÉCURITÉ

- i. Zero Trust
- ii. Passwordless



TECH RADAR





01

IA GÉNÉRATIVE.

- i. Développement augmenté par l'IA
- ii. Retrofit de la GenAI dans les apps



LES CONTRIBUTEURS DE CE CHAPITRE



**Soufiane
KELI**

AI apps Lead
soufiane.keli@octo.com



**Alain
FAURE**

Senior Architect & GenAI expert
alain.faure@octo.com



**Alexandre
JEAMBRUN**

CRAFT Community Practice Lead
alexandre.jeambrun@octo.com

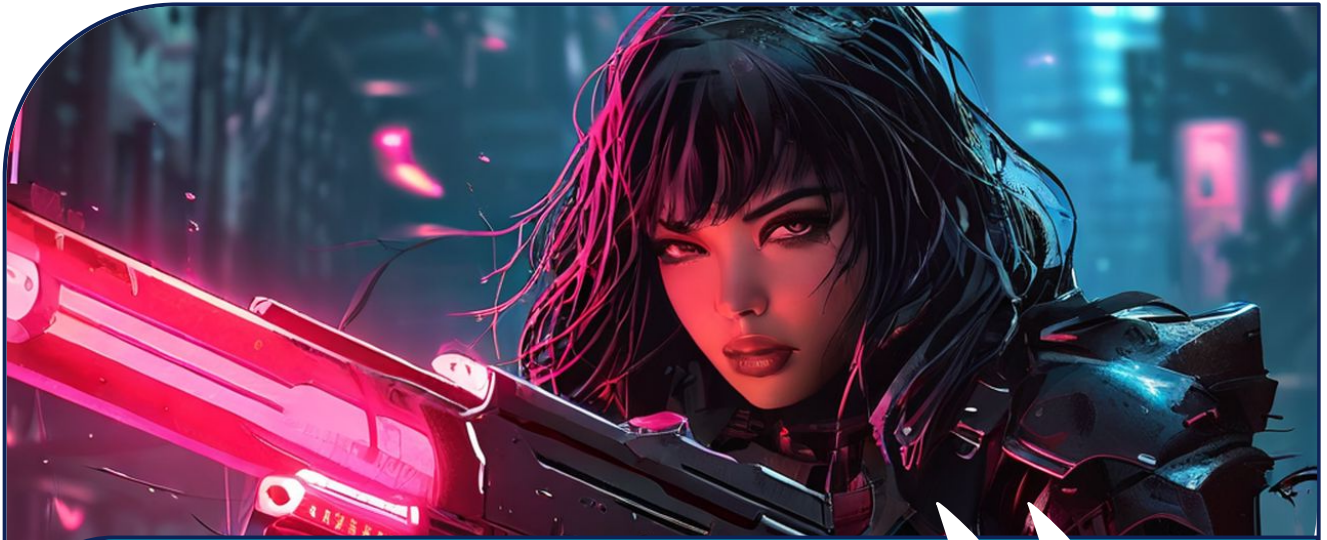


**Antoine
CHANTALOU**

Head of Software Engineering
antoine.chantalou@octo.com



01. IA GÉNÉRATIVE



GenAI :

“ Everybody be cool this is a robbery ! “

Il n'était pas envisageable d'entamer ce Tech Trends 2025 par un autre sujet que l'IA générative.

En effet, en 2024, l'**IA générative a réalisé un véritable hold-up**. La « *GenAI* » ouvre de nouvelles perspectives de valeur dans des domaines jusque-là peu touchés par l'automatisation.

Dans ce chapitre, nous traiterons les deux problématiques clés que pose la **GenAI** dans le monde du **Software Engineering** et qui nous impacteront en 2025 :

1. Le développement augmenté par l'IA
2. Le *retrofit* de la GenAI dans les applications

IA GÉNÉRATIVE

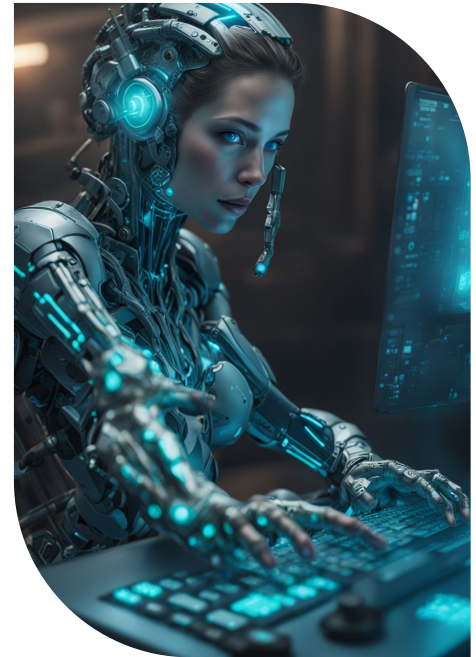


i. Développement augmenté par l'IA

i. Développement augmenté par l'IA

En moins de 2 ans, l'**IA générative** a ouvert de nouvelles perspectives de valeur dans différents domaines. L'écosystème du **Software Engineering** ne passera vraisemblablement pas entre les gouttes.

En 2025, les **assistants de code GenAI** auront gagné en maturité. Se pose alors une question brûlante : **l'intelligence artificielle a-t-elle un impact réel sur le développement logiciel ?**



L'émergence des **LLM**¹ et de la GenAI suscitent des réactions contrastées dans le monde du développement logiciel. Certains y voient une opportunité de réduire le temps et les efforts pour coder en déléguant les tâches fastidieuses ou répétitives, d'autres y voient une mode passagère. Et puis il y a ceux, qui comme Elon Musk, craignent que l'IA supplante l'humanité et appellent à un moratoire.

Nous verrons que la clé d'un déploiement réussi de ces outils réside dans le renforcement des bonnes pratiques de développement prônées par le **Software Craftsmanship**.

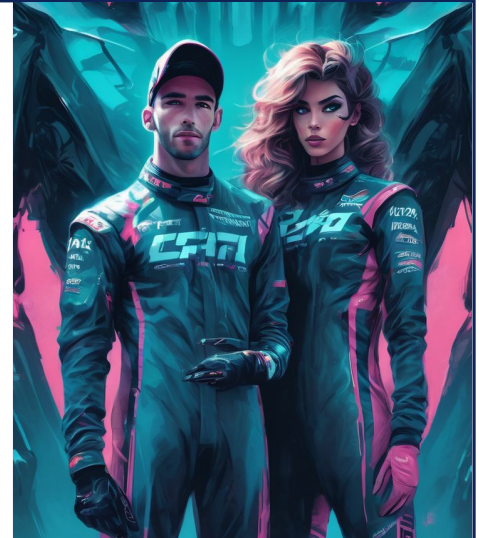
1. LLM (Large Language Model) : type de programme d'intelligence artificielle capable de reconnaître et de générer du texte. Ils sont entraînés sur des ensembles de données massifs.

Définition et cas d'usages

La **GenAI** pour les développeurs regroupe des outils intégrés aux IDE, comme : Github Copilot, Google Gemini, Tabnine, etc.

Ces outils proposent deux modes d'interaction pour la génération de code :

- **l'autocomplétion**
- et le **chat** avec **prompt**¹.



GenAI : avez-vous besoin d'un copilote ?

D'après nos retours d'expérience, la GenAI est pertinente pour :

- Générer de petits blocs de code, tests et documentation.
- Expliquer et refactoriser du code simple.
- Créer des applications simples ou des *POC* en phase d'innovation (réduisant le temps passé sur Stack Overflow).

Elle montre ses limites lorsqu'il s'agit de :

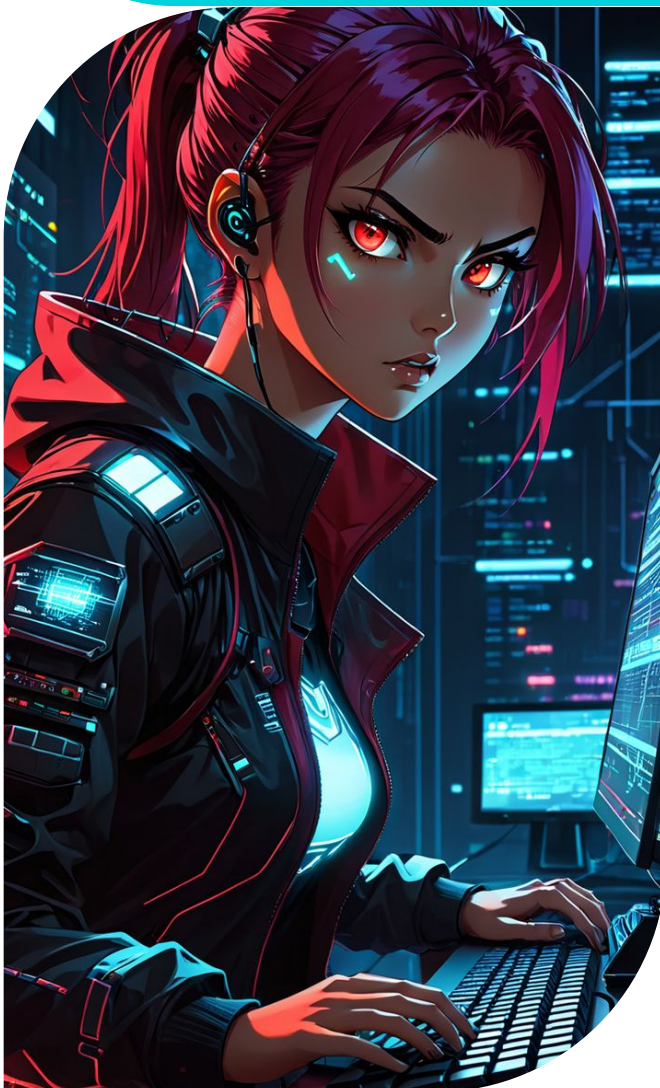
- Développer des applications complètes ou complexes.
- Faire évoluer une application existante.
- Refactoriser une application².
- Travailler sur les dernières versions de frameworks (où la base de code d'entraînement est limitée).

1. Prompt : une instruction fournie à un système d'IA, qui utilise ces informations pour générer des réponses ou des créations en texte, image, ou autre forme de média..
2. On commence à voir également des initiatives pour réécrire des bases de code legacy (par exemple dans le domaine de la Data science, de SAS vers Python).

Gain de productivité : entre mythe et réalité, ne succombez pas au discours marketing

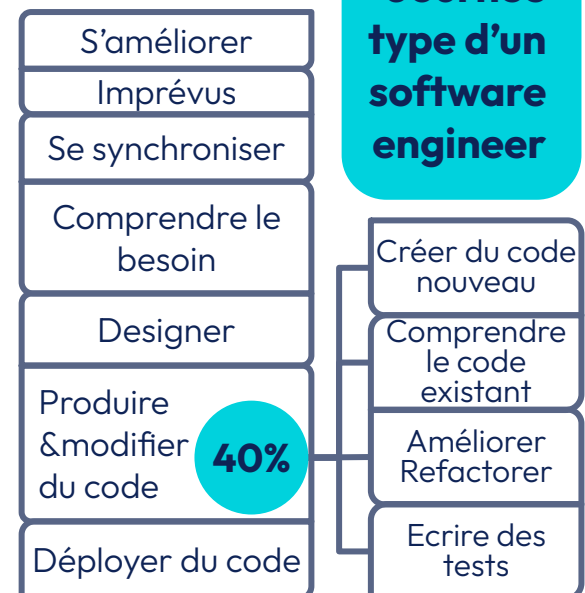


PRODUCTIVITÉ ?



Écrire du code représente au mieux 40 % du temps d'un développeur. Les gains générés par ces outils se limitent donc à cette portion.

Journée type d'un software engineer



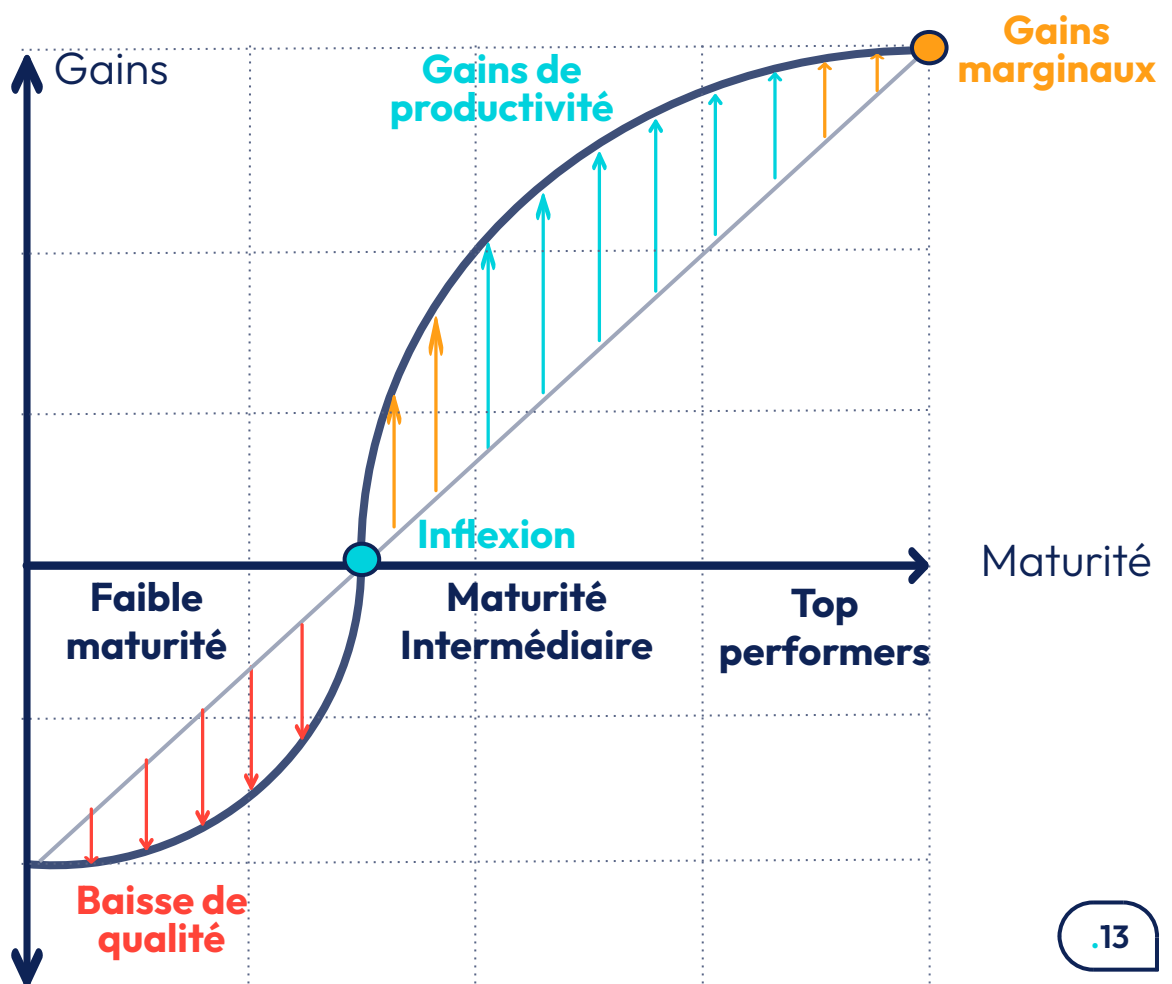
L'optimisation se situe entre 10% et 15% !

Source : étude "Today was a Good Day : The Daily Life of Software Developers"

Les gains de productivité ne sont pas équi-répartis



Notre expérience montre que les développeurs **de niveau intermédiaire à avancé en profitent le plus**, tandis que les experts en tirent peu d'avantages **et que les débutants sont pénalisés s'ils ne sont pas accompagnés**.

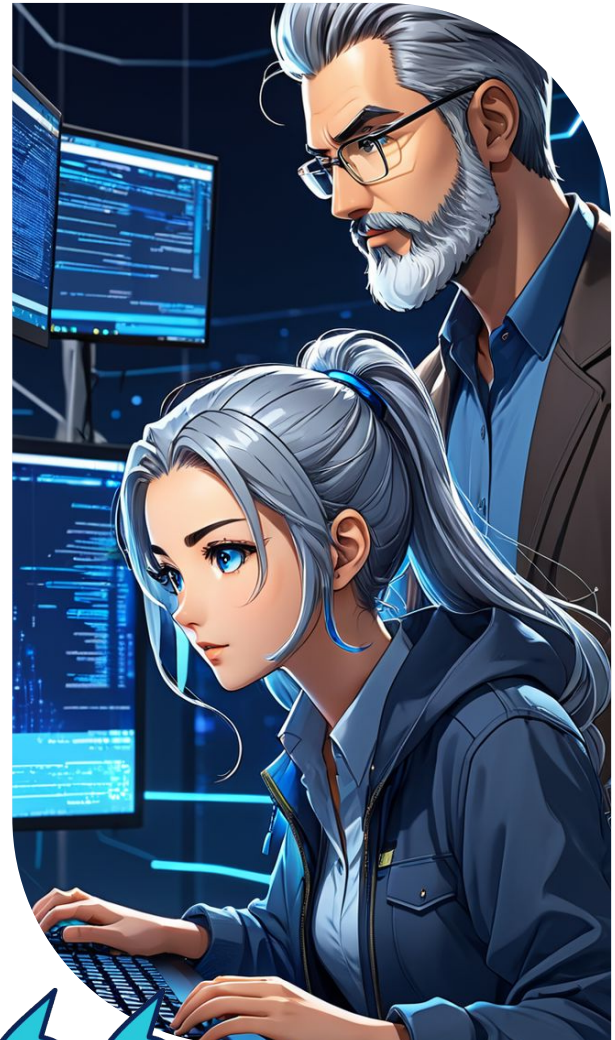


Attention à la pyramide de séniorité et au niveau de maturité de vos équipes

Un développeur inexpérimenté pourrait involontairement dégrader la qualité du code en se fiant aveuglément aux suggestions de l'IA. Et donc introduire des **hallucinations** dans la base de code.

En revanche, un développeur chevronné possède le recul et l'expertise nécessaires pour évaluer les propositions de l'IA, identifier les erreurs potentielles et les corriger¹.

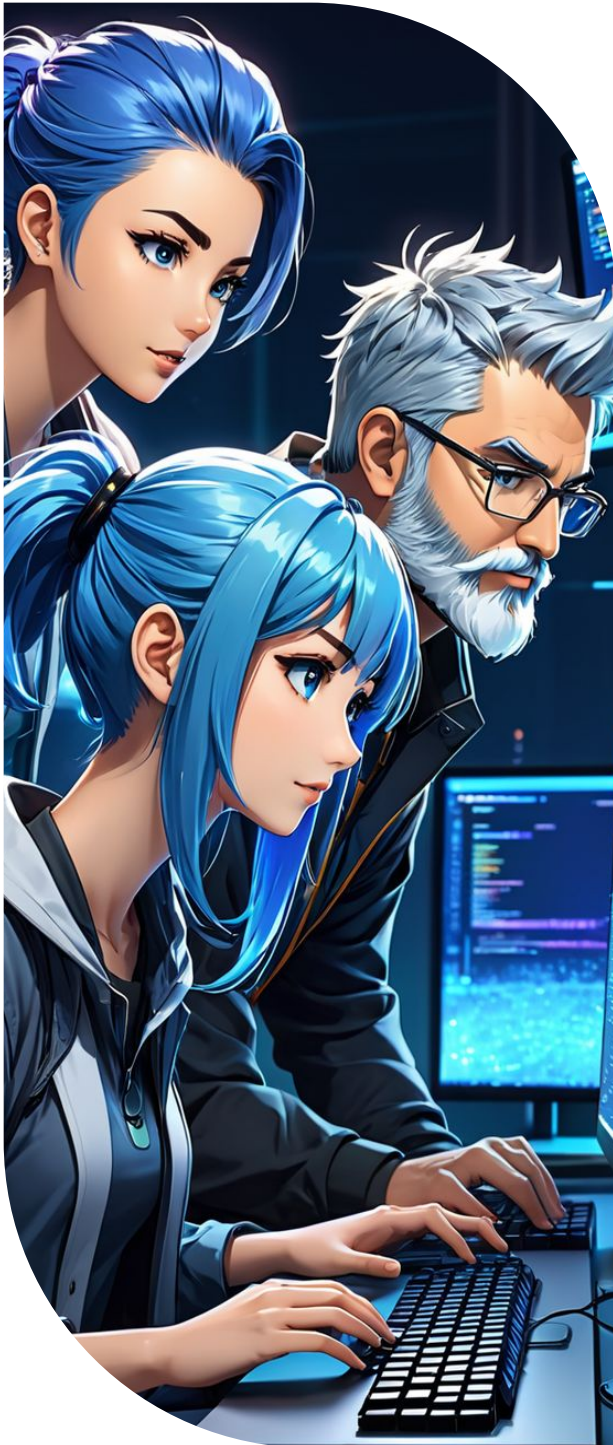
Les **développeurs seniors** sont essentiels dans vos équipes, pour déployer les pratiques du **Software Craftsmanship** (que nous abordons sur la page suivante) qui deviennent alors fondamentales, pour **sécuriser le déploiement de la GenAI**.



Les développeurs juniors doivent être accompagnés par les développeurs expérimentés

1. Sachant que des travaux comme [CriticGPT](#) d'OpenAI avancent sur la revue de code généré pour proposer des critiques qui mettent en évidence les inexactitudes dans les réponses.

Le CRAFT au chevet des assistants de code IA



La **culture CRAFT** s'avère essentielle pour un déploiement réussi de la GenAI. Ses **pratiques**¹ favorisent la **collaboration**, la **revue de code** et le partage des **connaissances**, permettant aux développeurs expérimentés de guider et de superviser l'utilisation de la **GenAI**.

Au-delà des pratiques, l'esprit du **compagnonnage** prôné par le CRAFT, le regard critique, la recherche de l'excellence et la remise en question permanente complètent et sécurisent bigrement le déploiement de la GenAI. En outre, le CRAFT est un rempart face au risque **d'hallucinations** des LLM.

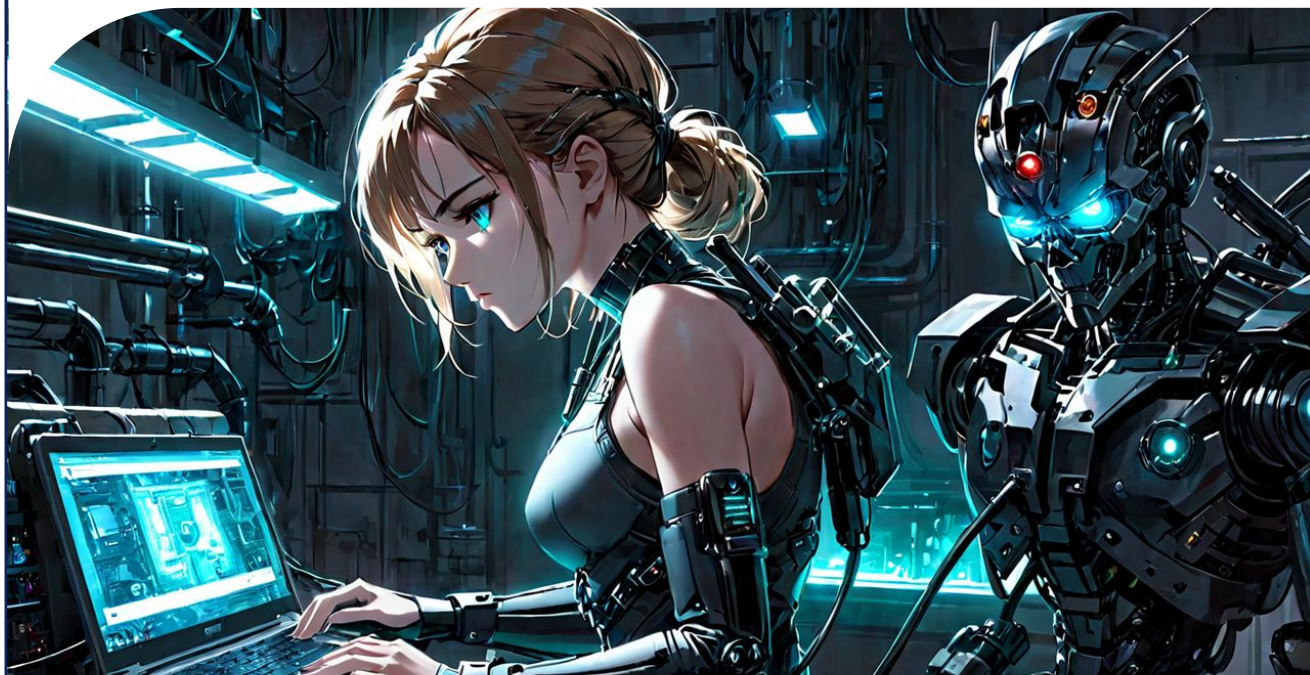
L'outillage dopé par la GenAI devient un allié pour renforcer les bonnes pratiques et améliorer la productivité.

1. Pair programming, Peer review, TDD (Test Driven Development), DDD (Domain-Driven Design), etc. Voir le détail dans cette [section](#).

Si les promesses de productivité sont le premier levier mis en avant, la GenAI nécessite de renforcer les bonnes pratiques de qualité de code.

Il est **crucial de plonger dans cette évolution technologique** avec un sens aigu du **CRAFT**. Cette transition doit s'accompagner d'une vigilance constante sur les implications **éthiques**, de **sécurité** (notamment de **souveraineté**) et de **gestion du changement**.

La GenAI pourrait encore creuser le fossé entre les organisations performantes qui ont investi sur la technologie et les développeurs... et les autres.



Alors en 2025, les assistants GenAI deviendront-ils des **outils indispensables** ou bien de **redoutables endosquelettes**¹ dont Sarah Connor devra nous débarrasser ? Ce qui est certain, c'est qu'ils continueront leur **percée** et proposeront très certainement une **efficacité accrue**.

1. Le T-800 est un cyborg conçu avec un châssis de combat hypersophistiqué avec un endosquelette recouvert de tissus biologiques humains. Il s'agit d'un prototype à l'origine créé par l'intelligence artificielle militaire Skynet. *The Terminator* (1984) — James Cameron.

IA GÉNÉRATIVE

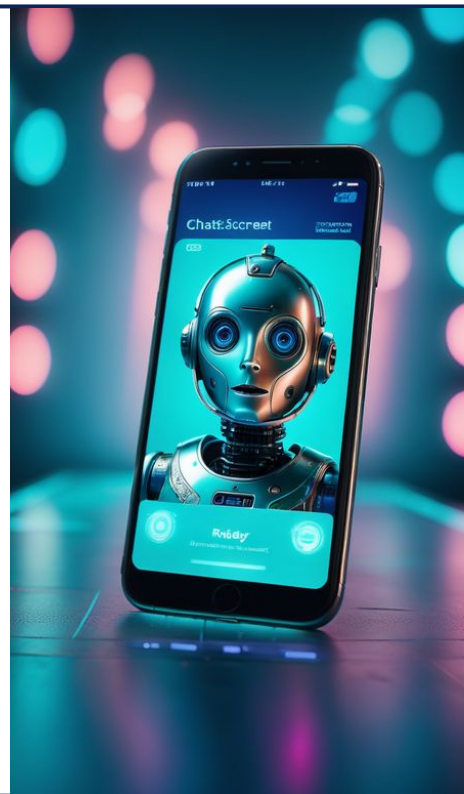


ii. Retrofit de la GenAI dans les apps

ii. Retrofit de la GenAI dans les apps

Avec le hold-up GenAI, l'IA passe **du sur-mesure au générique**. Plus besoin d'être un expert en deep learning pour **booster vos applications avec des fonctionnalités GenAI**.

Avec un **SI déjà « API-fié¹ »** et l'émergence de **GenAI providers « sur étagère »**, le retrofit de la GenAI dans les applications est à portée de main !



Concernant les aspects « produit » : il est nécessaire d'acculturer à la GenAI l'ensemble des acteurs. Les enjeux clés demeurent d'identifier les cas d'usages pertinents internes puis externes, et de sortir du cas d'usage *chatbot*.

Concernant les aspects « techniques » : un des enjeux clés est de choisir la méthode d'adaptation des LLMs appropriée, parmi les 4 existantes (que nous détaillons plus loin), pour ne pas faire exploser la complexité. Par ailleurs, le pattern d'architecture « AI Facade » permet de faciliter l'intégration des providers GenAI dans vos applications. Enfin, il convient de définir une stratégie de tests permettant de gérer des retours non déterministes.

1. APIfication : fait de lancer une stratégie pour exposer ses ressources et services numériques sous la forme d'APIs afin de faciliter la création de frontaux digitaux, de développer des partenariats et de nouveaux modèles d'affaires.

La recette magique pour intégrer la GenAI dans vos applications

Il existe certains points structurants à prendre en compte dès la phase de cadrage, sur les aspects produits et techniques.

Les enjeux du cadrage produit

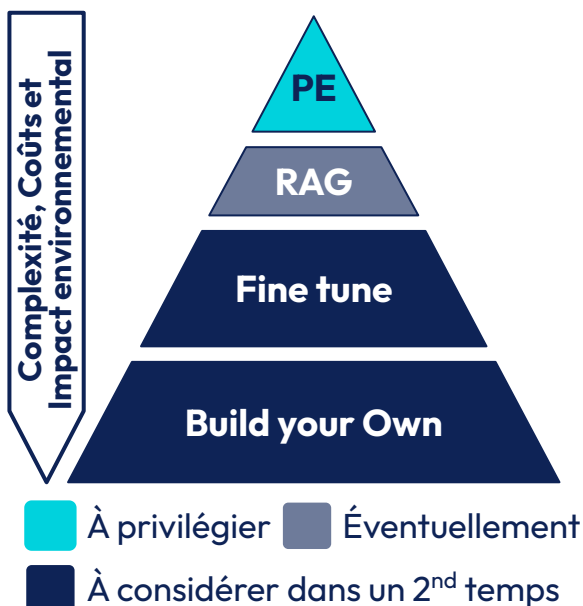
- ◉ **Acculturer** l'ensemble des acteurs non Data Scientists et les inciter à considérer les outils GenAI du marché : Designers, marketing, communication, les développeurs, les architectes, etc.) pour améliorer l'efficacité opérationnelle.
- ◉ **Identifier un portefeuille de use cases** : qui ne peuvent être réalisés sans GenAI et sortir du cas d'usage *chatbot* qui est peu innovant¹. Il est également important de bien comprendre les méthodes et coûts d'adaptation des LLMs (voir page suivante).
- ◉ **Combiner Greenfield & Brownfield** : bâtir une nouvelle application ou bien faire évoluer une application existante. L'amélioration d'un produit existant avec un usage de la GenAI en retrofit est souvent dévaluée (voir notre section « *Brownfield* » pour la reprise d'un asset existant).



1. L'idée est de sortir de l'interaction et rentrer dans des use cases où l'on exploite la capacité de « réflexion » du GenAI, avec l'utilisation de fonctions avancées : traitement en plusieurs étapes (avec plusieurs LLMs), chain-of-thought, agents, etc.

Les enjeux du cadrage technique & architecture

- **Méthodes d'adaptation des LLMs** : privilégier les méthodes PE ou RAG, qui répondent à une grande majorité de use cases, et qui sont facilement accessibles..



Prompt engineering (PE) : injection de texte dans le prompt afin d'orienter le modèle vers des réponses désirées.

Retrieval Augmented Generation (RAG) : enrichissement du contexte à partir de sources de données complémentaires.

Fine tune : entraînement en profondeur d'un modèle de langage pré-entraîné sur un ensemble de données plus spécifiques.

Build your Own : construction et entraînement de son propre modèle sur un large ensemble de données.

- **AI Facade** : intégrer les GenAI providers via le pattern *façade* pour réduire le couplage, puisque le marché n'est pas encore stabilisé.
- **DATA** : veiller à la qualité des données pour augmenter la fiabilité des résultats, notamment si vous mettez en place un RAG qui nécessite des données correctement qualifiées et indexées dans une base vectorielle.
- **Tests** : définir une stratégie de tests pour gérer des retours non déterministes.

Le Prompt Engineering : la pierre angulaire pour interagir avec les modèles GenAI

Le **Prompt Engineering** (ingénierie de la requête) consiste à injecter du texte dans le *prompt*, afin d'orienter le modèle vers des réponses désirées. C'est devenu un élément clé pour tirer parti de la puissance des **LLM** et **LMM**¹.

La qualité du **prompt** a un **impact déterminant** sur la **pertinence des réponses** du modèle. Le PE permet également de réduire considérablement le **coût d'usage** de ces modèles.



Que le prompt soit rédigé par un utilisateur, ou codé par un développeur (ou un mix des deux), les bonnes pratiques de Prompt Engineering sont **répertoriées** ([ici pour OpenAI par exemple](#)) : *préciser les consignes, le contexte, le format de sortie, donner des exemples, etc.*

Néanmoins il n'y a **pas de recette magique** pour garantir un bon résultat. De ce fait, un **jeu de tests** ainsi **qu'une méthode d'évaluation** des réponses fournies par le modèle **sont nécessaires** pour évaluer la pertinence des réponses, potentiellement différentes à chaque exécution.

1. LLM (Large Language Model) : type de programme d'IA capable de reconnaître et de générer du texte ; entraînés sur des ensembles de données massifs. LMM (Large Multimodal Model) : type de programme d'IA capable traiter en input et output plusieurs types de contenus (texte, images, son).

Contrôle de la pertinence des réponses du modèle

Malgré l'attention portée sur le Prompt Engineering, il est nécessaire d'anticiper le fait que le **modèle ne retourne pas une réponse pertinente** ou même exploitable par l'application.

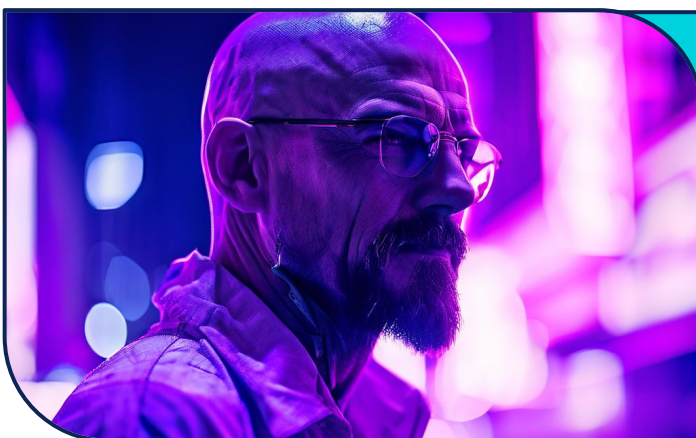
L'application doit donc comporter une **étape de validation** de la réponse du modèle à partir de contrôles :

- **syntaxiques** (la réponse est bien formée ?) ;
- **heuristiques** (les valeurs des champs sont plausibles ?).

L'application pourra alors décider si elle peut exploiter le résultat, ou si elle doit débrayer sur une approche alternative. Par exemple, une recherche par mot clé de type *Elastic*, une invitation à l'utilisateur de reformuler sa demande, etc.

En synthèse :

- La création du prompt est une étape fondamentale. Sans capacité d'évaluation il est impossible de savoir si le prompt est de qualité.
- La pertinence du résultat fourni par le modèle doit être évaluée et contrôlée par l'application avant de l'exploiter.



“Never Give Up Control”

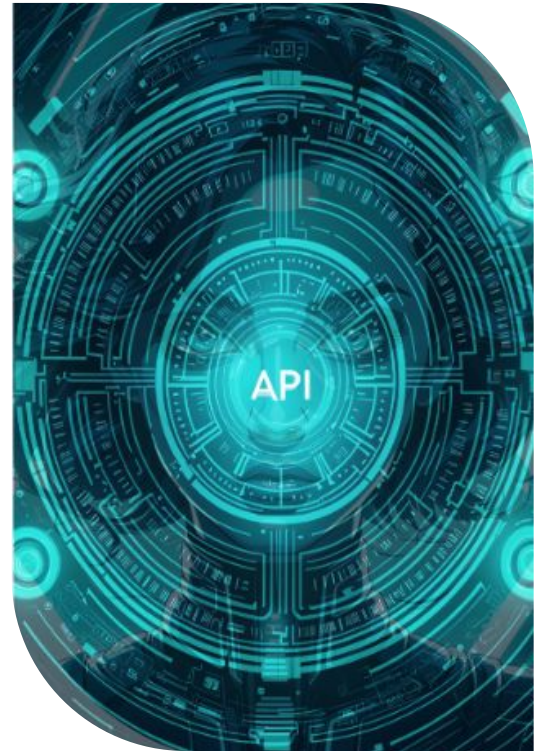
— Walter White, Breaking Bad, season 5 episode 8

AI facade : intégrez les modèles GenAI avec un couplage faible pour une meilleure évolutivité

En terme d'**architecture**, puisque le marché des providers GenAI n'est pas encore stabilisé, nous préconisons d'utiliser un composant **AI Facade** dont le rôle est de **découpler** les traitements GenAI du reste de l'application.

En outre, ce composant permet :

- de pré-traiter les requêtes de gestion de prompt ;
- d'appeler les services GenAI ;
- de post-traiter les réponses ;
- d'orchestrer les interactions avec les autres systèmes.



Le découplage offre divers avantages :

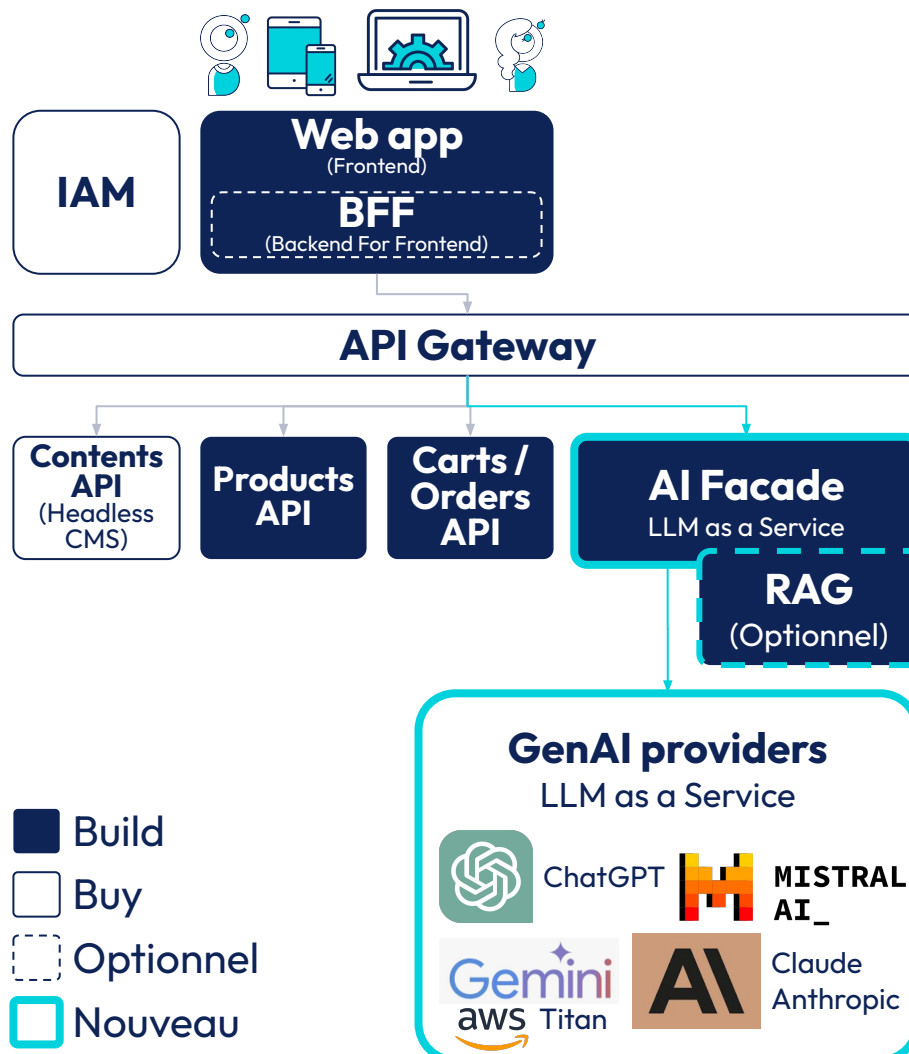
- réutilisabilité ;
- indépendance des cycle de développement et déploiement ;
- possibilité de désactiver le composant GenAI en cas de problème critique sans impacter le reste du site ;
- possibilité de tester ce composant de manière autonome.

A minima, prévoir un design modulaire pour se préparer à un découpage ultérieur (on parle de « majestic monolith »).

Exemple d'architecture GenAI

Le schéma suivant illustre l'impact de l'introduction de la GenAI sur une architecture e-commerce type, via la méthode PE.

- Il s'agit essentiellement d'un chantier de **développement** et **d'intégration**.
- **Pas de révolution** si on a déjà lancé son chantier d'**ouverture** & **API-sation du SI** avec les aspects **sécuritaires** afférents.

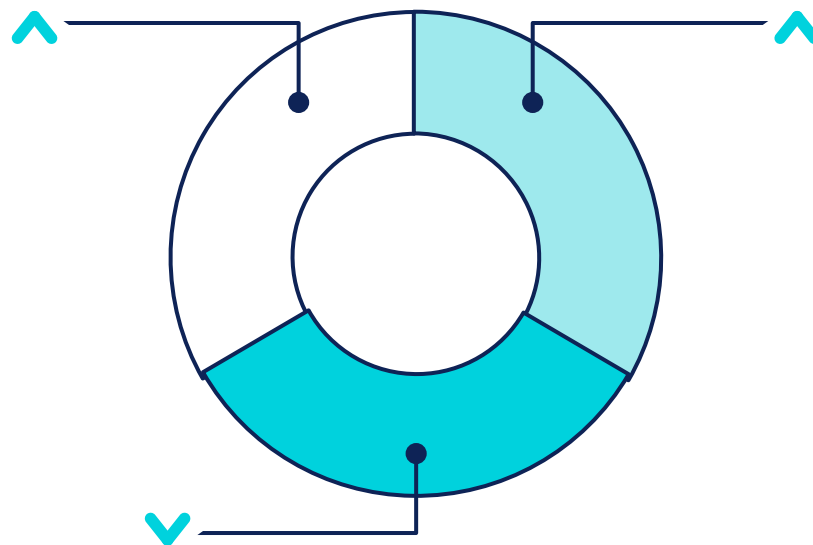


Quelques types de GenAI providers

Si aujourd'hui *OpenAI* domine le marché, plusieurs alternatives existent pour se doter des capacités de GenAI.

Les **fournisseurs spécialisés** comme *Anthropic* ou *Mistral*. Ces pure players proposent des solutions plus spécialisées (texte, image, vidéo) et peuvent offrir des coûts moins élevés ou des modèles de tarification plus flexibles.

Les **cloud providers** comme *Azure*, *AWS*, *GCP* ont renforcé leurs offres avec des capacités d'IA générative. L'intégration de ces nouveaux services à leur plateforme permet d'avoir une offre complète et une facturation sur le même compte de votre organisation.



Les **marketplaces open source** comme *Hugging Face* ou *PyTorch Hub* packagent et distribuent des modèles open source variés pour traiter du texte, des images, de la voix ou des vidéos.

Conclusion

L'intégration de la GenAI dans un produit existant via la **méthode PE n'est pas plus complexe** que celle d'un autre service. Ce n'est pas une révolution de votre SI ou de vos équipes : c'est **avant tout un projet d'intégration**. Attention néanmoins à la complexité et aux coûts entraînés par les autres méthodes¹.

Par défaut, nous recommandons d'**utiliser des services GenAI sur étagère**. Si l'utilisation d'API GenAI ne requiert pas de connaissances approfondies en data science, il est nécessaire de sensibiliser vos équipes au **PE**, aux risques d'**hallucinations** et à une stratégie de **tests** pour gérer des retours non déterministes.



Alors en 2025, comme **Paul Atréides**², considèrerons-nous les **GenAI providers** comme la nouvelle **ÉPICE** : la ressource la plus convoitée pour **transformer vos application par l'IA** (en marge d'octroyer des dons de *prescience* et de rendre possible les voyages interstellaires) ?

1. La méthode RAG nécessite la construction d'une base vectorielle qui requiert une expertise *document engineering*. Le *fine-tuning*, pourrait augmenter avec les SLMs (Small Language Models) plus performants dans certains domaines. *Build your Own* n'est pas recommandé.

2. Paul Atréides (Timothée Chalamet) héritier de la Maison Atréides, s'enfuit dans le désert d'Arrakis, où il est recueilli par les Fremen. *Dune: Part One* (2021) — Denis Villeneuve.



02

ARCHITECTURE.

- i. Architectures composables
- ii. Architectures continues



LES CONTRIBUTEURS DE CE CHAPITRE



**Antoine
CHANTALOU**

Head of Software
Engineering

antoine.chantalou@octo.com



**Pierrette
BERTRAND**

Web-front Engineering
Lead

pierrette.bertrand@octo.com



**Adrien
GRAUX**

Software Architecture
Lead

adrien.graux@octo.com



**Soufiane
KELI**

AI apps Lead

soufiane.keli@octo.com



**Sylvain
FAGNENT**

Senior consultant
Architect

sylvain.fagnent@octo.com



**Alain
FAURE**

Senior Architect
GenAI expert

alain.faure@octo.com



**Pierre-Jean
DOUSSET**

Architecture Practice &
Security Lead

pierre-jean.dousset@octo.com



**Armen
OZCELIK**

API Lead

armen.ozcelik@octo.com



02. ARCHITECTURE



« L'architecture est le point de départ lorsqu'on veut conduire l'humanité vers un avenir meilleur. »
— Le Corbusier

Concernant l'architecture, l'année 2025 comportera 2 enjeux phares :

1. Bâtir des **architectures composables** afin de répondre aux enjeux de flexibilité et d'évolutivité de vos systèmes, avec l'essor de la « *MACH architecture* » qui tire notamment la mise en oeuvre d'une stratégie « *Going headless* ».
2. Déployer le principe de **Continuous architecture** qui consiste à transformer le rôle de l'architecte pour faire évoluer et grandir - sans rupture - les systèmes informatiques au fur et à mesure des besoins avérés, avec un prisme terrain.

ARCHITECTURE



i. Architectures composables

i. Architectures composables

Définition

Bâtir des architectures **composables**, c'est créer des **systèmes flexibles et adaptatifs** où les composants peuvent être assemblés et réassemblés — comme un système LEGO® — pour répondre à des besoins spécifiques, sans nécessiter de reconstruction complète ou de modifications majeures.

Le saint Graal tant espéré des DSI pour architecturer et simplifier les évolutions de leur SI !



Photo by Xavi Cabrera on Unsplash

Pourquoi maintenant ?

Le contexte socio-économique sous tensions oblige à une **efficience opérationnelle accrue**. Les architectures composables permettent de répondre aux **enjeux croissants d'agilité et d'innovation** en offrant plusieurs caractéristiques :

- Le découpage en **composants modulaires** et interopérables.
- La **scalabilité** technique et organisationnelle et l'évolutivité.
- Le **découplage** technologique et adapté aux usages et la possibilité de combiner une approche « **best of breed** ».



Photo by Marko Blažević on Unsplash

“Build **small** systems that are **replaceable** over **big** system that are **maintainable**, better it is.”

Les architectures composables sont donc la promesse de pouvoir facilement **remplacer ou faire évoluer des parties du système**, plutôt que d’avoir à maintenir ou faire évoluer un système global complexe.

Mais au-delà du concept, penchons-nous sur une implémentation concrète : la MACH architecture !

MACH Architecture : “ One architecture to rule them all “ ?



Photo by Matthew Ball on Unsplash

En 2025, les **architectures MACH** vont certainement continuer de gagner en popularité et méritent qu'on s'y frotte.

Ce style d'architecture vise à concevoir des systèmes **composables**, en prônant 4 principes fondamentaux :

- ◉ **Microservices,**
- ◉ **API-first,**
- ◉ **Cloud-native,**
- ◉ **Headless.**

S'ils existent depuis longtemps (POO ~1980, SOA ~2000, API REST ~2000-2010, microservices ~2014, cloud computing ~2016, etc.), nous assistons aujourd'hui à une **stabilisation des technologies et des écosystèmes** sur la base de ces quatre principes fondamentaux.

L'acronyme « MACH » a donc le vent en poupe et a le mérite de poser une vision d'architecture dont l'ambition est d'être la plus « **future-proof** » possible en bâtissant des **systèmes composables, évolutifs et scalables**.

Nous en détaillons ci-après les 4 piliers.

Microservices

Définition

Les microservices sont des composants applicatifs **autonomes** qui portent sur un **périmètre fonctionnel délimité**, et qui mettent en jeu des **silos verticaux** sur **trois couches** :

- des **interfaces** (exposition via API ou événements),
- des **règles de gestion** métier,
- et une solution de **persistance** indépendante.

Ils sont pertinents dans les contextes de fort besoin de **modularité** et/ou **d'indépendance des équipes**.



Les 3 promesses phares

1. **Scalabilité** : découper un monolithe qui ne « scale » plus, pour permettre de délivrer fréquemment, sans régressions, et améliorer le Time to Market (T2M).
2. **Découplage organisationnel** : permettre de faire évoluer un système en limitant les impacts sur les autres systèmes.
3. **Découplage technique** : permettre à chaque composant de tirer parti des technologies spécifiques qui répondent à ses usages.

Les 3 points de vigilance

1. **Maîtrise des systèmes distribués** : l'approche microservices, par sa nature distribuée, induit une complexité tech. & orga. et requiert une maturité des équipes sur le **Software Craftsmanship**, l'**Agilité** et le **DevOps**.
2. **Entropie du SI** : les architectures microservices repoussent la complexité dans l'intégration interservices et doivent donc être mises en œuvre **à partir d'une certaine complexité**. Il est préférable de partir d'un monolithe modulaire pour éprouver le découpage (on parle de « **majestic monolith** »), puis si la complexité le justifie, de découper en microservices. [Avez-vous remarqué que l'acronyme *MACH* fonctionne aussi avec « *Majestic monolith* » ? 🧸]
3. **Granularité** : éviter un micro-découpage des systèmes qui accroît anormalement la complexité. La bonne granularité est celle d'un découpage par **domaines métier** (Domain Driven Design) avec une **API documentée**.



Granularity ?

“In general, microservices should cleanly align to bounded contexts.”

— Sam Newman,
Building Microservices

API-first

Définition

Le terme API (Application Programming Interface) désigne une interface normalisée par laquelle une application offre des services à une autre application.

Aujourd'hui, un très grand nombre d'entreprises proposent un accès à une partie de leur système d'information via des APIs sur le web (en pratique, souvent des APIs REST ou GraphQL) de manière sécurisée via HTTPS, OAuth2 et OpenID Connect.



Le développement d'une API porte essentiellement sur la **logique d'exposition des ressources métiers et le design de l'API** (en simplifiant : conception des requêtes et des réponses et des formats de données). Les développeurs peuvent alors créer des applications basées sur ces APIs. On parle de **mashup** quand ces applications reposent sur la composition de plusieurs APIs, provenant éventuellement de systèmes disjoints.

L'UI (User Interface) peut ainsi être développée par des équipes plus expertes dans l'implémentation des parcours utilisateurs, de l'ergonomie et de l'accessibilité numérique.

Les 3 promesses phares

1. **100% API coverage** : pour concevoir des systèmes modulaires et interopérables. Avec une approche « *API-first* », il ne suffit pas d'exposer, via des APIs, les fonctionnalités nécessaires à la couche de présentation (UI), comme dans le cadre d'une démarche Headless, mais **d'offrir une couverture API totale qui couvre l'intégralité des fonctionnalités métier du système**.
2. **DYOC** : le pattern « *Drink Your Own Champagne* » fait que l'on est soi-même utilisateur de son API pour garantir que l'on vive la même expérience que celle que l'on offre à ses clients.
3. **L'ouverture du SI** : faire émerger de **nouveaux modèles d'affaires** en bâtissant une API **générique**, agnostique des consommateurs et en **industrialisation la consommation** de cette API.



1. "I Want to Hold Your Hand" est une chanson des Beatles, écrite par John Lennon et Paul McCartney. Enregistrée le 17 octobre 1963, elle apparaît sur leur premier disque enregistré en utilisant un équipement 4 pistes et en stéréo.

Les 3 points de vigilance

1. Ne pas **réduire la stratégie d'API-sation à l'achat d'une solution d'API Management**, qui n'est pas un « *golden hammer* ». Les véritables enjeux à traiter sont : le design technico-fonctionnel des APIs, la sécurisation et la stratégie de découpage avec les impacts sur l'architecture et l'organisation.
2. Ne pas foncer tête baissée sur le pattern « **API Facade** », qui permet d'effectuer des transformations limitées. Les services exposés par les back-end sous-jacents laissent en général transpirer leurs défauts et contraintes (antipattern « *lipstick on a pig* »).
3. Ne pas faire que des APIs : proposer une documentation et un auto-enrollment qui permet **d'industrialiser la consommation des APIs** mesurée par le TTFAC (Time to First API Call) qui doit être de l'ordre d'une dizaine de minutes.

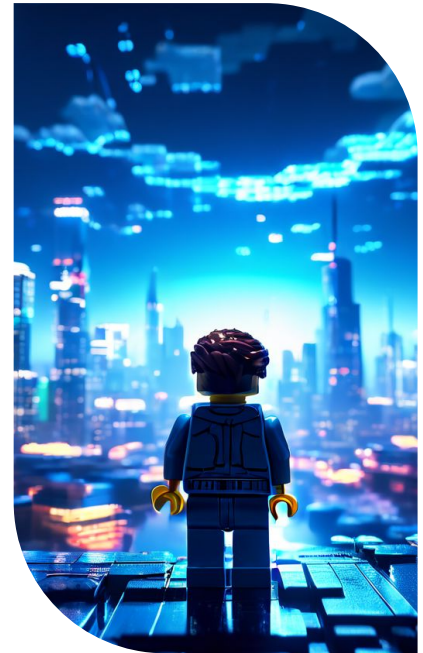


Photo by Daniel K Cheung on Unsplash

Cloud-native

Définition

Une application cloud-native est conçue dès le départ pour exploiter pleinement les capacités du cloud et en respectant les principes des twelve-factor app. Elle tire parti des services managés et des avantages inhérents au cloud computing, tels que la scalabilité, la résilience, l'élasticité et l'intégration CI/CD.



Copyright @OCTO Technology

Les 3 promesses phares

1. **Évolutivité et flexibilité** : l'élasticité du cloud permet à l'infrastructure d'évoluer en fonction des besoins.
2. **Résilience et haute disponibilité** : les applications cloud-native reposent sur des microservices et des conteneurs qui améliorent l'isolation et le découplage.
3. **Accélération du T2M** : les services managés réduisent la complexité opérationnelle et permettent de se concentrer sur la partie applicative. Les pratiques DevOps favorisent des phases de test et de déploiement plus rapides et plus fréquentes.

Les 3 points de vigilance

L'accélérateur cloud et ses services managés sur étagère sont devenus une nécessité dans de nombreux cas pour répondre aux enjeux d'agilité, d'accélération du *Time to Market* et d'innovation. Néanmoins, une approche holistique est nécessaire, notamment pour intégrer les aspects :

- de **maîtrise des coûts** et **volumétrie** (désamorcer via une stratégie *FinOps* une facture cloud qui peut grimper en fonction du trafic) ;
- de **l'expertise** des équipes in situ sur le **DevOps** et les **Hyperscalers** ;
- de **souveraineté** au regard des données manipulées et du contexte métier, et de risque de **cloud lock-in**.



Cloud ?

« Il est plus facile de monter dans le cloud que d'en descendre »

— Inconnu, les pieds sur terre.

Headless

Définition

« *Le Headless, c'est couper la tête !* »

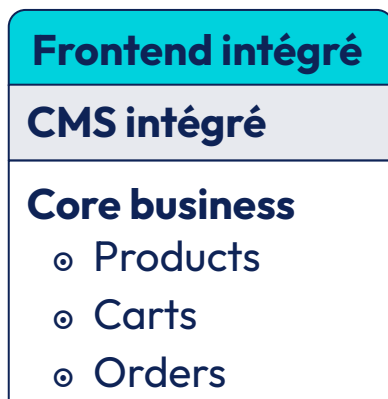
L'architecture Headless consiste à séparer la couche de présentation de la couche business et contenu, via APIs.

Un chantier « *Going headless* » consiste à passer d'un système monolithique à un écosystème “ *best of breed* ” pour permettre de livrer de manière indépendante les composants front et back, et d'offrir des expériences premium aux utilisateurs sur l'ensemble des touchpoints et canaux digitaux.



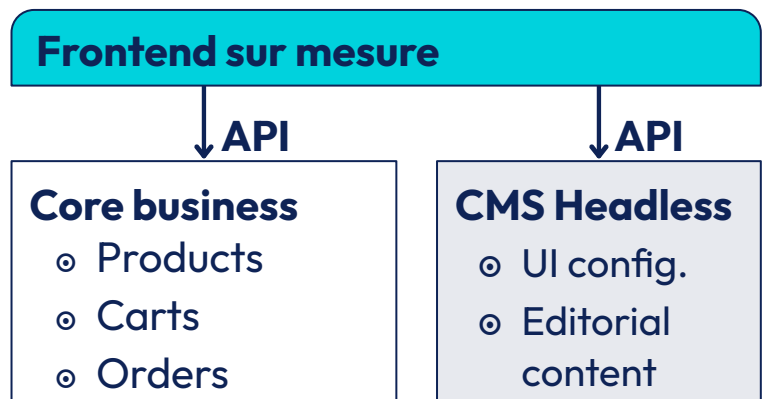
Architecture existante

Solution vendor monolithique intégrée.



Architecture headless

Le **cœur de métier** reste dans la solution vendor et est exposé via **API**. Le **CMS** est déporté pour être interopérable via **API**.



Les 3 promesses phares

- ◉ **ATAWAD** : en séparant la couche présentation des couches business et contenu par des APIs omnicanales et agnostiques du terminal, nous apportons une réponse concrète aux enjeux *Anytime, Anywhere, Any Device*. Cela nous permet de délivrer une expérience cohérente sur tous les touchpoints, où que soient nos utilisateurs, en concevant autant de frontaux digitaux que nécessaire qui consomment une même couche API.
- ◉ **Offrir des expériences premium dans un monde où les paradigmes UI évoluent à vitesse grand V** : front-end web, applications mobiles, devices IoT, voire des assistants conversationnels LLM basés sur de l'IA générative.
- ◉ **Amélioration du T2M et de l'agilité** : améliorer la réactivité et la souplesse pour répondre aux besoins business souvent marketing.

Photo by Artur Tumasyan on Unsplash



Les 3 (et quelques) points de vigilance

- ◉ Pour la partie **frontend**
 - > Privilégier une UI *responsive* et *mobile first* basée sur une *bibliothèque de composants*.
 - > Intégrer les enjeux de *l'accessibilité numérique* (voir chapitre dédié).
 - > Intégrer les contraintes *SEO* et de *webperf* (en utilisant les patterns d'architecture adaptés : CSR, SSR, PWA).
- ◉ Pour la partie **CMS Headless**
 - > La migration de données depuis le CMS intégré.
 - > La formation des contributeurs marketing sur les nouveaux workflow de publication.
- ◉ Pour la partie **Core business**
 - > L'exposition des données et services via des APIs (voir la [section API-first](#)).



Photo by Daniel K Cheung on Unsplash

Architecture pragmatique

Si l'on fait le choix de définir un cadre d'architecture (notamment pour donner une vision), aligner les acteurs et minimiser l'empilement des technologies et trends qui éclosent au fil des saisons, il est fondamental de savoir rester pragmatique : se laisser une liberté dans les choix techniques en fonction des besoins qui émergent, tout en sachant sortir du cadre quand cela est nécessaire.

Appliquer sans discernement les principes des architectures MACH sans tenir compte du contexte peut être contre-productif et amener à des solutions trop complexes, ou trop tôt, au regard des besoins réels.

En particulier, si vous vous faites le choix de l'architecture MACH, nous insistons sur trois mises en gardes, que nous détaillons ci-après.

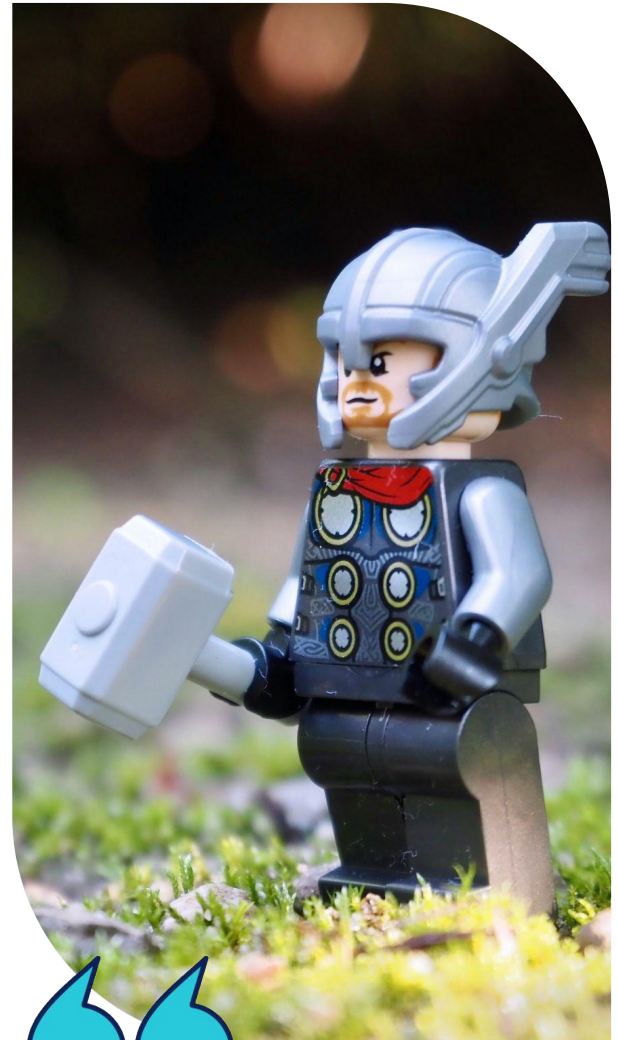


Photo by Mateusz Wactawek on Unsplash

« Si le seul outil que vous avez est un **marteau**, vous tendez à voir tout problème comme un **clou** ».

— Abraham Maslow

MACH : comment rester simple et ne pas se perdre dans la complexité ?



I love simplicity.
— Zendaya

Majestic monolith first

Démarrer avec un monolithe, quitte à le découper plus tard, est une approche pragmatique, qui est notamment recommandée par Martin Fowler.

Granularité et DDD¹

Lorsque l'on découpe son système en sous-systèmes, la question de la granularité est primordiale. Nos retours d'expérience ont démontré qu'il fallait à tout prix éviter un hyper-découpage par services, avec une granularité trop fine. La « *bonne granularité* » est celle du domaine métier.

EDA² vs API

Il arrive parfois que le pattern API-first ne soit pas la solution la plus adaptée. Une architecture événementielle peut être plus pertinente pour certains cas d'usages³.

1. DDD : Domain-Driven Design.

2. EDA : Event Driven Architecture.

3. Notons que l'inverse est également vrai : l'approche Event Driven, popularisée par la démarche event storming, ne doit pas non plus être systématisée !

Conclusion

En 2025, les architectures MACH continueront de prendre de l'ampleur. Ce style d'architecture a précisément l'ambition de concevoir des **systèmes composables** et propose une **vision d'architecture** basée sur quatre patterns d'architecture éprouvés : *Microservices*, *API-first*, *Cloud-native* et *Headless*.

Cette architecture peut apporter des bénéfices et donner un cap, si l'on sait **rester pragmatique** en sachant éviter le **piège** d'un micro-découpage et en se laissant la liberté de **sortir du cadre** quand cela est pertinent.



Alors en 2025, comme **Maverick**¹, passerez vous à la vitesse **MACH 10** avec la **MACH Architecture** ?

1. Pete « *Maverick* » Mitchell (Tom Cruise) est un aviateur avec un tempérament de « *chien fou* », peu apprécié de sa hiérarchie et marqué par la disparition en vol de son père. *Top Gun* (1984) — Tony Scott.

ARCHITECTURE



ii. Architectures continues

ii. Architectures continues

Définition

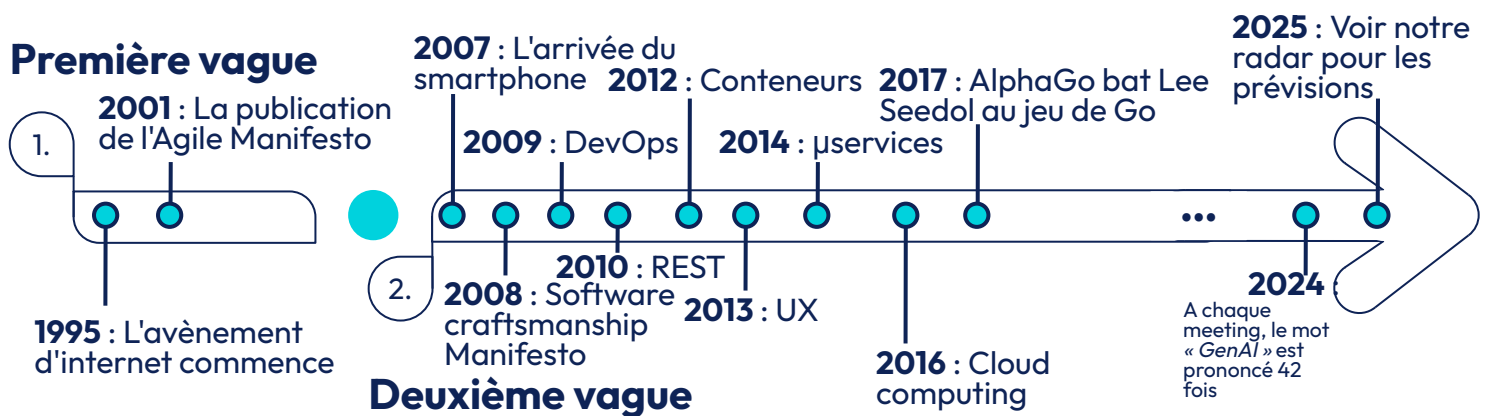
La “ *Continuous Architecture* ” consiste à faire évoluer et grandir sans rupture les systèmes informatiques au fur et à mesure des besoins avérés.

Du point de vue de la construction d'un produit, cela consiste à avoir une **vision d'une architecture cible** qui colle à celle de la vision produit, pour partir dans la bonne direction tout en **posant des principes d'architecture haut niveau**. Ensuite, il s'agit de définir une stratégie technique et de faire évoluer l'architecture en continu, tout en faisant évoluer la vision technique et la vision produit.



De la complexité IT maîtrisée à l'ère du mode produit

La trajectoire de l'informatique a été marquée par des innovations et des changements de paradigmes qui ont profondément modifié notre approche de la construction et de la gestion des systèmes d'information.



À la fin de la première vague :

- La complexité restait maîtrisable.
- Se projeter sur le long terme restait possible au travers de schémas directeurs.
- Travailler en mode projet reproductible restait acceptable.

2008 : l'arrivée du smartphone révolutionne l'accès à l'information et les interactions numériques.

Désormais, suite à la deuxième vague :

- La complexité accélère en même temps que les besoins IT.
- Les besoins deviennent fortement transactionnels (en opposition aux besoins statiques du début) pour le grand public.
- Le trafic internet explose.
- 2016 : l'explosion du cloud computing révolutionne l'intégration avec des services tiers.



“ Software is eating the world.
In the future every company will become
a software company.”

— Marc Andreessen

La deuxième vague souligne la **transition d'un monde où la planification à long terme était possible à une ère où l'adaptabilité et la réactivité sont devenues essentielles** pour répondre aux besoins changeants et croissants en technologie.

Dans la section suivante, nous détaillerons **l'impact de la continuous architecture sur l'organisation et les pratiques.**

L'ère du produit a bunkerisé des architectes dans leur tour d'ivoire

Avec l'essor de l'agilité, de par la proximité des TL¹ avec le produit, des architectes se sont retrouvés plus éloignés des décisions et ont pu perdre en autorité. En effet, la rédaction de DATs² très détaillés, comprenant parfois même des exemples d'implémentation, est un processus long dont le cycle ne correspond plus aux exigences de réactivité et de flexibilité.

Les TL ont alors porté à la fois les sujets de software design, mais aussi ceux liés à l'architecture (et parfois avec un manque de recul).

Le *software design* est lié aux caractéristiques du code (lisibilité, organisation, maintenabilité, etc). Alors que l'*architecture* veille à la robustesse, la scalabilité et la performance sur le plan fonctionnel, technique et infrastructure.



Le TL est devenu un mouton à 5 pattes !

1. TL : le *Tech Lead* ou *Lead Developer* est celui qui, dans une équipe produit, porte la vision technique et en est garant.

2. DAT : Dossier d'Architecture Technique.

La clé est une collaboration fine entre Architecte et Tech Lead



Nos expériences montrent qu'il est nécessaire de définir un périmètre de responsabilités clair.

L'architecte :

- ◉ **oriente** et aide à la décision en particulier dans les choix d'une meilleure intégration au SI ;
- ◉ **freine** l'entropie du SI ;
- ◉ fait profiter les équipes de son **expérience**.

Le TL :

- ◉ **contribue** à la décision ;
- ◉ porte la **qualité** et la **maintenabilité** du produit d'un point de vu logiciel ;
- ◉ fait profiter les équipes de sa **connaissance produit** et son expertise.

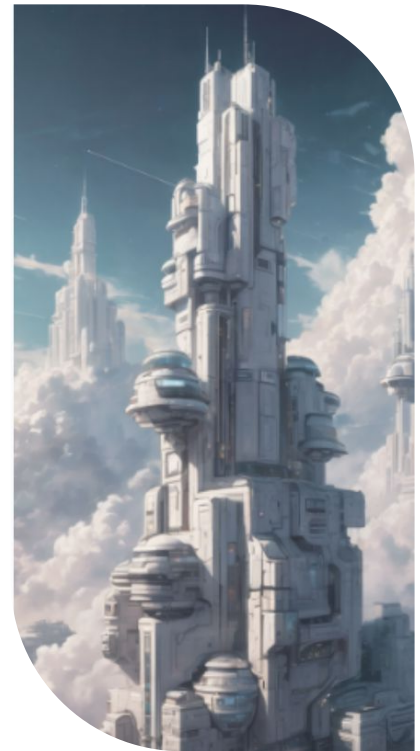
La symbiose de l'architecte et du Tech Lead peut ainsi fonctionner pour le bénéfice du produit.

Les rituels comme vecteur de l'architecture continue

Il faut à tout prix éviter l'anti-pattern de l'architecte **tour d'ivoire** et développer la collaboration en continu via des rituels :

- les Tech Leads sont partie prenante des **rituels d'architectures** quand les problématiques les concernent.
- les architectes sont à la disposition des équipes de développement et sont invités aux **rituels des équipes**.

Notons que la clé de la collaboration repose également sur les capacités de l'architecte à fédérer et à développer une légitimité (techniques, soft skills).

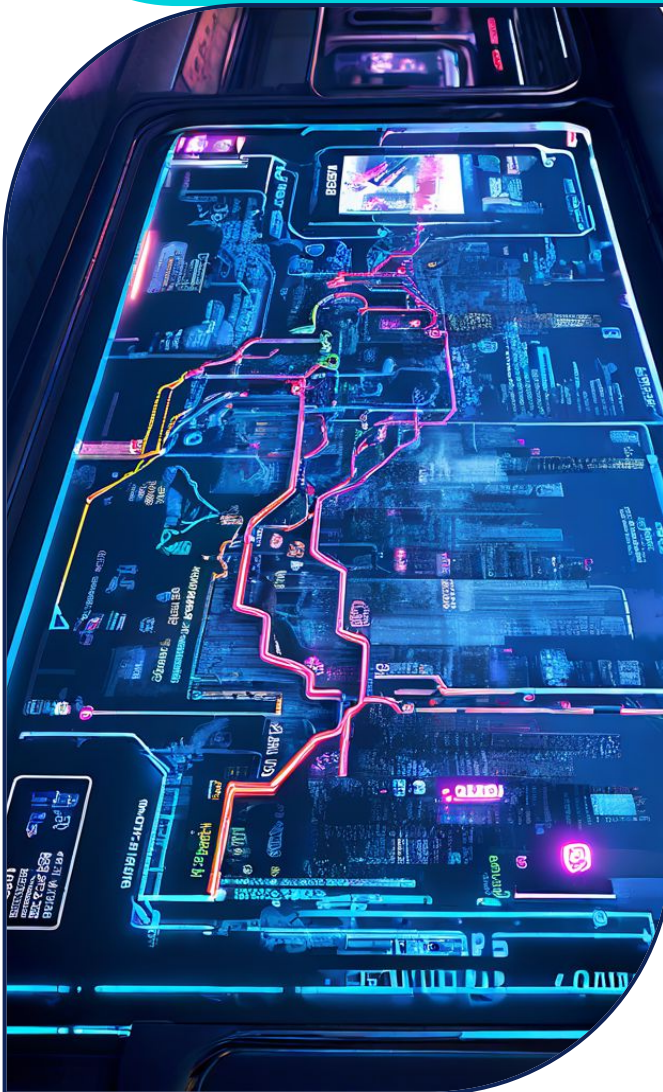


Les points d'attention clés de l'architecte

- **Stratégie technique** : orientation technique, garant de la cohérence générale du SI dont la conception, garant du timing de décision des évolutions majeures.
- **Gouvernance de conception** : revues de conception et d'architectures.
- **Coordination entre équipes** : coordination et intégration des composants pour assurer une cohérence globale.
- **Performances en production** : robustesse des architectures, observabilité.

Vers l'automatisation de la cartographie applicative

La cartographie applicative aide au dialogue entre les différents acteurs et permet d'évaluer les impacts SI.



Historiquement, pour maintenir une cartographie applicative à jour, il fallait généralement s'y atteler tous les deux ans. Mais dans notre contexte actuel, les cartographies réalisées manuellement deviennent rapidement obsolètes.

L'automatisation de la cartographie s'avère donc indispensable.

Aujourd'hui, les outils d'APM¹, avec la fonction de service map, se révèlent être d'une **aide précieuse** pour obtenir une **cartographie dynamique** à partir des flux de production, mais ils ne la remplacent pas.

1. Outils d'APM : *Application Performance Monitoring* qui permet aux organisations de superviser la performance de leurs applications. Exemple : *Elastic apm*, *Grafana*, *Datadog*, etc.

L'Architecture Decision Record : clé de voûte pour l'alignement

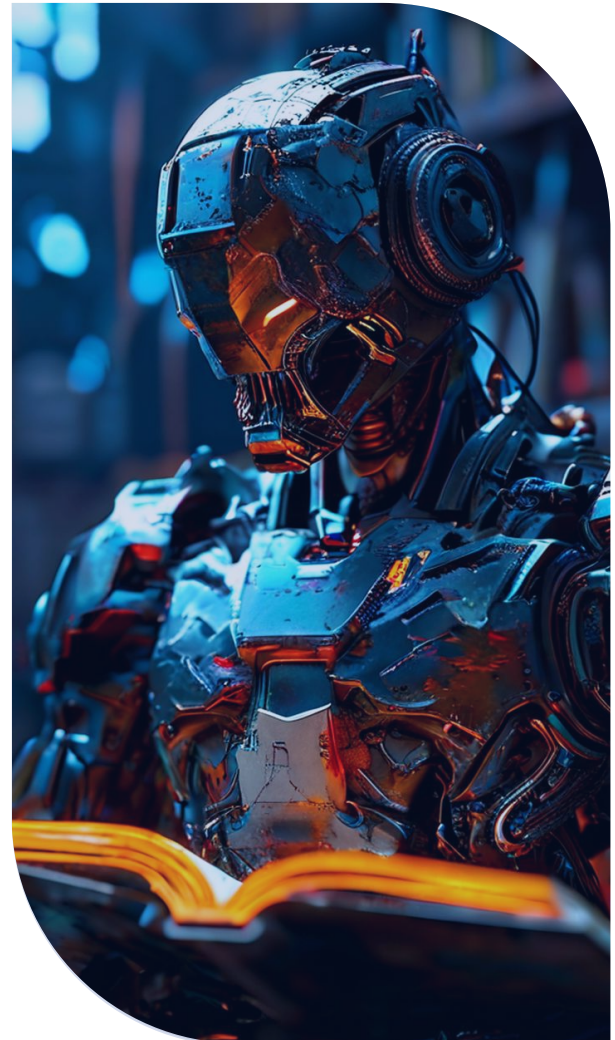
Avec une forte accélération des changements, les décisions aux mains des équipes autonomes, le turnover, ou encore les équipes distribuées en remote, il est devenu primordial d'aligner les acteurs en **formalisant les choix d'architecture**.

La difficulté réside dans le fait de documenter avec la bonne granularité, en se restreignant aux **choix d'architecture** dans leurs **concepts** et dans leurs **principes**.

Exemples :

- les critères d'un découpage applicatif ;
- l'architecture d'intégration avec un système tiers ;
- les SLAs, suivant les types d'interaction.

L'ADR s'impose aujourd'hui comme le document immuable, durable et facilement consultable.



ADR : responsabilité de l'architecte, qui entérine les décisions avec les parties prenantes : TL, Engineering manager, Senior Staff Engineer, etc.

Nous évoluons dans une ère où l'adaptabilité et la réactivité sont devenues essentielles.

La **Continuous Architecture** souligne l'importance de créer une **collaboration fine et continue** entre les architectes et les Tech Leads.

La Continuous Architecture peut impacter l'organisation. La gouvernance peut être adaptée pour renforcer la collaboration autour des décisions d'architecture, via des **rituels**.

L'**automatisation de la cartographie** du SI devient primordiale.

L'**ADR** devient alors clé pour l'alignement.



Alors en 2025, comme le Dr. Emmett Brown¹, provoquerez-vous une rupture du **continuum espace-temps** en sautant dans la **Continuous Architecture** ?

1. Le docteur Emmett Brown, dit « Doc » (Christopher Lloyd) est l'inventeur de la machine à voyager dans le temps, sous la forme d'une DeLorean qui fonctionne au plutonium. *Back to the Future* (1985) — Robert Zemeckis.



03

EFFICIENCE OPÉRATIONNELLE & SUSTAINABLE.

- i. erooM
- ii. Brownfield
- iii. Accessibilité numérique
- iv. Développement Hybride



LES CONTRIBUTEURS DE CE CHAPITRE



**Tristan
NITOT**

Communs &
Anthropocène
tristan.nitot@octo.com



**Sébastien
BARTOLI**

Sustainable Software
Lead
sebastien.bartoli@octo.com



**Antoine
CHANTALOU**

Head of Software
Engineering
antoine.chantalou@octo.com



**Alexandre
JEAMBRUN**

CRAFT Community
Practice Lead
alexandre.jeambrun@octo.com



**Soufiane
KELI**

AI apps Lead
soufiane.keli@octo.com



**Nicolas
LAURENT**

Mobile Lead
nicolas.laurent@octo.com



**Romy
DUHEM-VERDIÈRE**

Accessibility Lead
romy.duhem-verdiere@octo.com



Armen OZCELIK

API Lead
armen.ozcelik@octo.com



03. EFFICIENCE OPÉRATIONNELLE & SUSTAINABLE

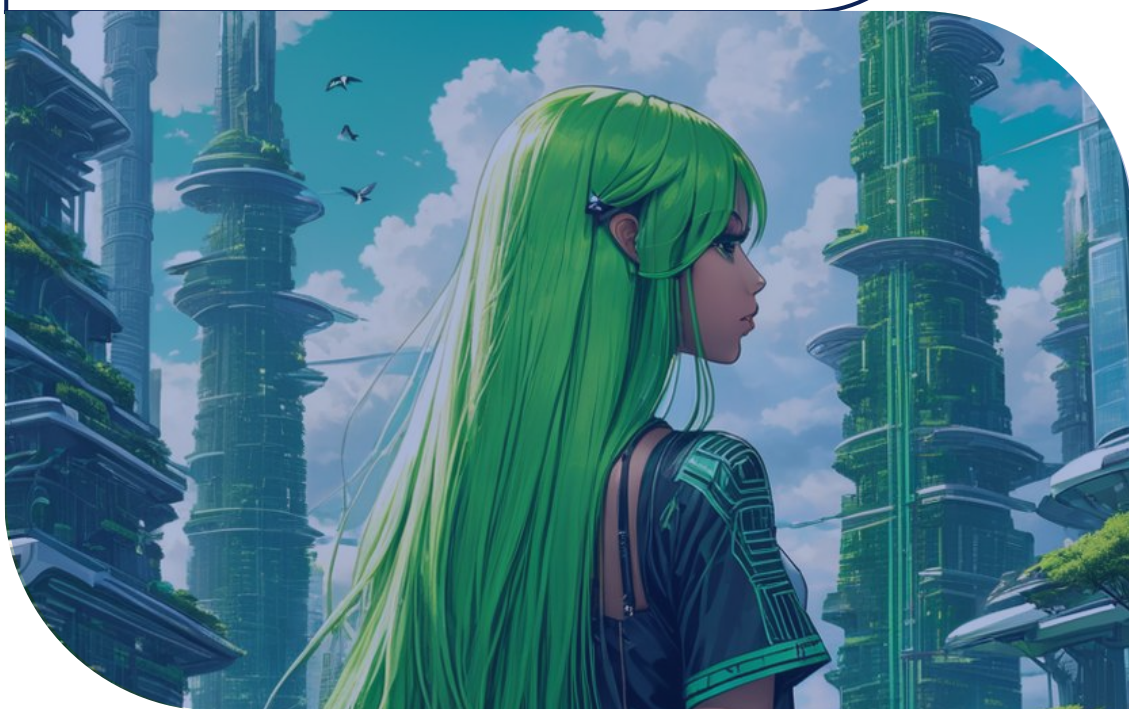


Sustainability et régulation IT : les alliés inattendus de votre efficacité opérationnelle.

Le contexte socio-économique sous tension que nous avons connu en 2024 induit la nécessité d'améliorer notre **efficacité opérationnelle**. L'année 2025 portera de forts enjeux sur le **Sustainable** et la **régulation IT**. Nous détaillons 4 tendances :

1. De la Loi de Moore à Loi d'**erooM**.
2. L'approche **Brownfield**, pour recycler et reprendre le contrôle de nos assets.
3. Dans un contexte de stabilisation des écosystèmes techniques, le **développement hybride** est devenu un incontournable.
4. Étant désormais passible de sanctions, l'**accessibilité numérique** n'est plus une option en 2025.

EFFICIENCE OPÉRATIONNELLE & SUSTAINABLE



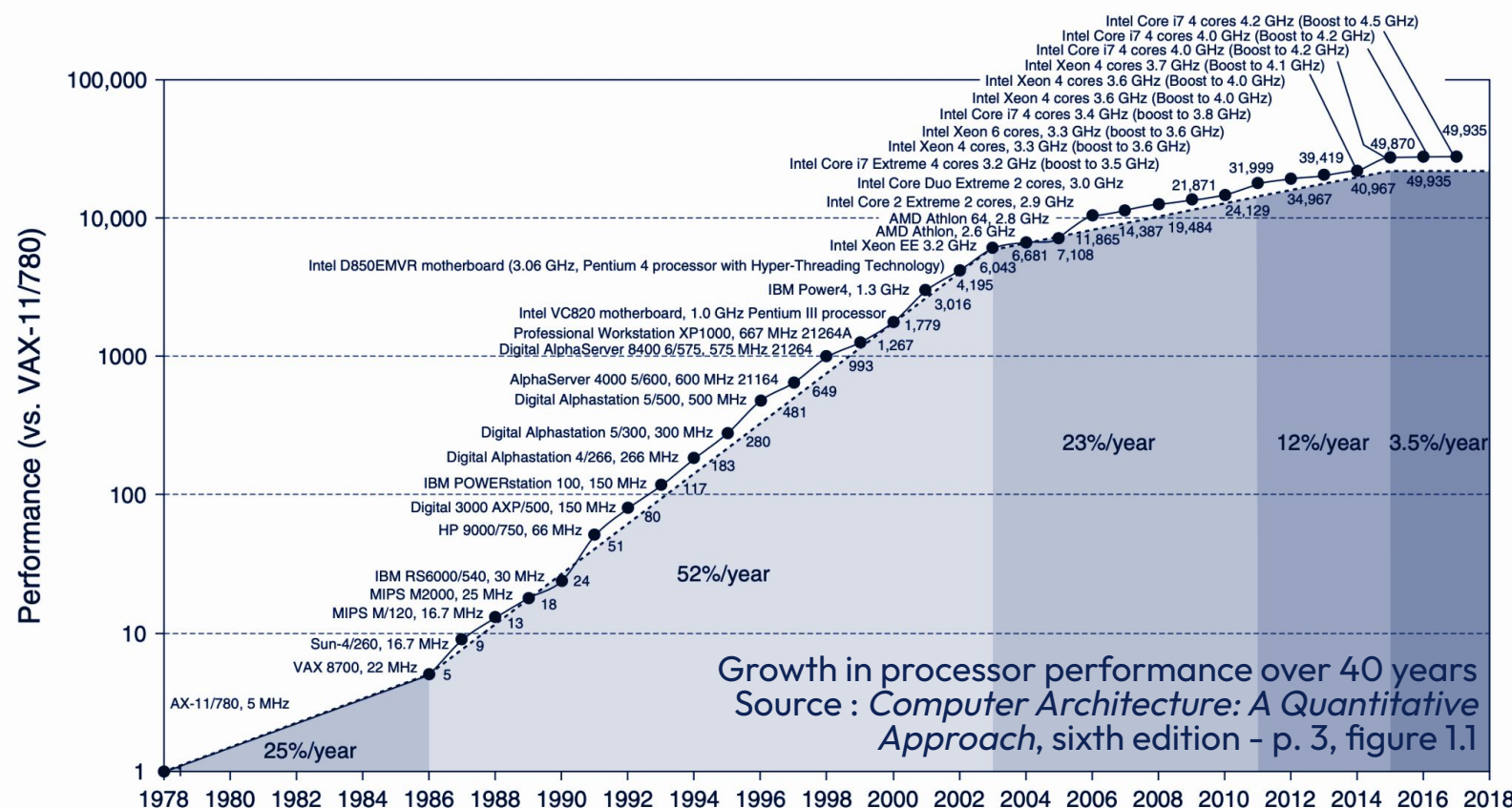
i. erooM

i. De la Loi de Moore à Loi d'eroomM

Gordon Moore, co-fondateur d'Intel, avait énoncé en 1965 : « *le nombre de transistors dans un semiconducteur double tous les 2 ans* ».

Mais aujourd'hui, la puissance de calcul ne suit plus la prédiction de Moore et nous assistons en parallèle à une raréfaction des ressources matérielles.

En effet, au risque de surprendre, d'après le graphe *Growth in processor performance over 40 years*, ce n'est plus une réalité : **la loi de Moore est morte.**



La loi d'erooM

On pourrait paniquer face à la fin de la loi de Moore. Mais c'est en fait une opportunité. En effet, depuis que la loi de Moore existe, on a pu valider l'existence de la loi de Wirth, qui stipule que les logiciels ralentissent aussi vite que le matériel accélère. Comme on disait à une époque : « *ce qu'Intel vous donne, Microsoft vous le reprend* ».

Dans ce contexte, Tristan Nitot, fondateur de Mozilla Europe et défenseur du logiciel libre, nous enjoint avec la loi d'erooM à optimiser nos logiciels pour inventer de nouveaux usages, tout en consommant moins de ressources, moins d'énergie... et tout en continuant à fournir du travail à l'industrie du logiciel.



« La loi d'**erooM** (Effort Radicalement Organisé d'Optimisation en Masse) nous enjoint à **optimiser nos applications et systèmes d'un facteur 2 tous les 2 ans** »

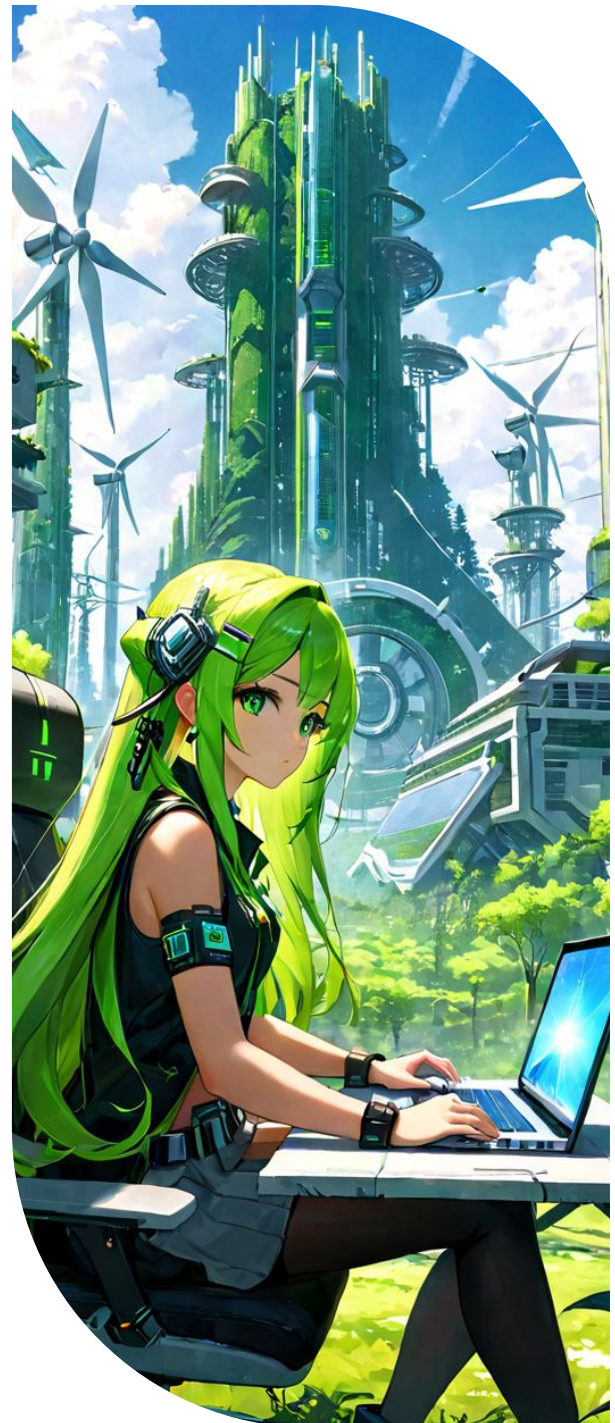
Tristan Nitot, fondateur de Mozilla Europe et défenseur du logiciel libre

Les bénéfices d'erooM

Si l'objectif principal de la loi d'erooM consiste à économiser la ressource numérique, elle nous permet également de réduire de manière drastique **l'empreinte environnementale de notre SI**.

Les effets bénéfiques sont les multiples :

- ◉ **optimiser ses coûts** par la réduction des infrastructures, moins sollicitées par un logiciel plus performant ;
- ◉ **améliorer l'UX** et les taux de conversion en réduisant le poids des pages, les besoins en CPU et en RAM, et les temps de chargement ;
- ◉ **réduire notre empreinte environnementale** en limitant le rachat de matériel ;
- ◉ **fidéliser les développeurs**, en donnant du sens.



Pourquoi maintenant ?



Nous savons que les ressources nécessaires à la fabrication de nouveaux terminaux s'amenuisent et que l'empreinte carbone du numérique continue d'augmenter.

C'est pourquoi l'État commence à légiférer. Le **RGESN**¹, publié dans sa seconde version le 17 mai 2024, compte 78 critères qui devraient s'imposer par décrets dans les prochaines années.

D'autre part, dès aujourd'hui, toutes les entreprises de plus de 500 salariés ont l'obligation de réaliser leur bilan carbone tous les 4 ans et de **publier un plan de transition pour réduire leurs émissions de gaz à effets de serre**. Ceci s'accompagne aussi d'un véritable enjeu d'image et chacun observe un véritable mouvement de fond.

1. RGESN : Référentiel général d'éco-conception de services numériques.

erooM induit une démarche d'éco-conception

Pour accompagner cette transition vers loi d'erooM, il convient de s'appuyer sur le RGESN qui promeut l'**éco-conception** des services numériques et vient compléter les référentiels numériques responsables existants : RGPD et RGAA (dont nous parlons plus loin).

Bien entendu, il convient de mettre en place cette démarche dans le processus de delivery, afin que les optimisations soient réalisées **au fil de l'eau**.

Nous recommandons de commencer par un assessment RGESN pour obtenir une première vue stratégique. Ce premier exercice, qui représente en moyenne 2h de mobilisation des leads techniques et fonctionnels d'un produit selon nos retours d'expérience, nous permet d'identifier les grands points sur lesquels se concentrer.



Une démarche du global au local

Il est possible de s'appuyer, en complément du RGEN, sur les outils suivants pour avoir une vue plus détaillée :

- sur l'analyse des usages et le ressenti utilisateur avec des outils comme *Matomo*, pour identifier les fonctionnalités les plus utilisées et qu'on aura le plus intérêt à optimiser ;
- sur la mesure de la **performance ressentie** via des outils comme les *Core Web Vitals* ;
- sur l'identification des éléments de code les plus consommateurs en temps et en ressources, avec des outils comme les *flame charts* pour vraiment entrer dans le détail.



Dans la suite du chapitre, nous verrons que ce contexte erooM est propice à d'autres tendances, notamment l'approche Brownfield.

La contrainte en tant que levier d'optimisation



Alors que le matériel disponible sur un ordinateur grand public de 2024 est désormais environ **1 million de fois plus puissant** que dans les années 60, les logiciels continuent eux à être toujours plus gourmands en ressources.

Puisque la puissance est disponible, il faut s'imposer des contraintes pour optimiser correctement.

Commencez par vous fixer comme limite haute les valeurs moyennes dans l'industrie actuellement. C'est une première marche qui devrait être tout à fait atteignable. Par exemple :

- ◉ 2.4 MB comme poids maximal pour une page web ;
- ◉ 40 requêtes HTTP.

Avec plus de maturité sur les sujets d'optimisation, on peut continuer à diviser par 2 sur les indicateurs, ou en ajoutant des critères comme la consommation de RAM ou en abaissant les versions d'OS supportées. Finalement, la seule limite est votre imagination.

Des architectures plus frugales



On l'a vu plus haut, les architectures *headless* sont l'occasion parfaite de tester des modèles par essence plus légers, tel que *Islands Architecture*.

Dans ce modèle, on génère en avance et on sert des contenus majoritairement statiques, que l'on vient alimenter de façon ciblée avec des composants dynamiques. Bénéficiant ainsi des excellentes performances du HTML statique, tout en pouvant utiliser ses APIs et proposer des expériences riches et composables.

Conclusion

La loi de Moore est morte. Tristan Nitot nous enjoint avec la **loi d'erooM** à **optimiser nos applications** et **systèmes** d'un **facteur 2 tous les deux ans**, pour consommer moins de ressources et inventer de nouveaux usages.

Pour accompagner cette transition, il convient de s'appuyer sur le **RGESN** qui promeut l'**éco-conception** des services numériques. Une part importante porte sur les enjeux de *web perf.* C'est aussi l'occasion de mettre en œuvre des architectures front plus frugales comme *Islands Architecture*.



Alors en 2025, osez-vous comme **Oh Dae-su**¹ affronter — un par un — les obstacles induits par la raréfaction des ressources, en embrassant les paradigmes d'**erooM** et de l'**éco-conception** ?

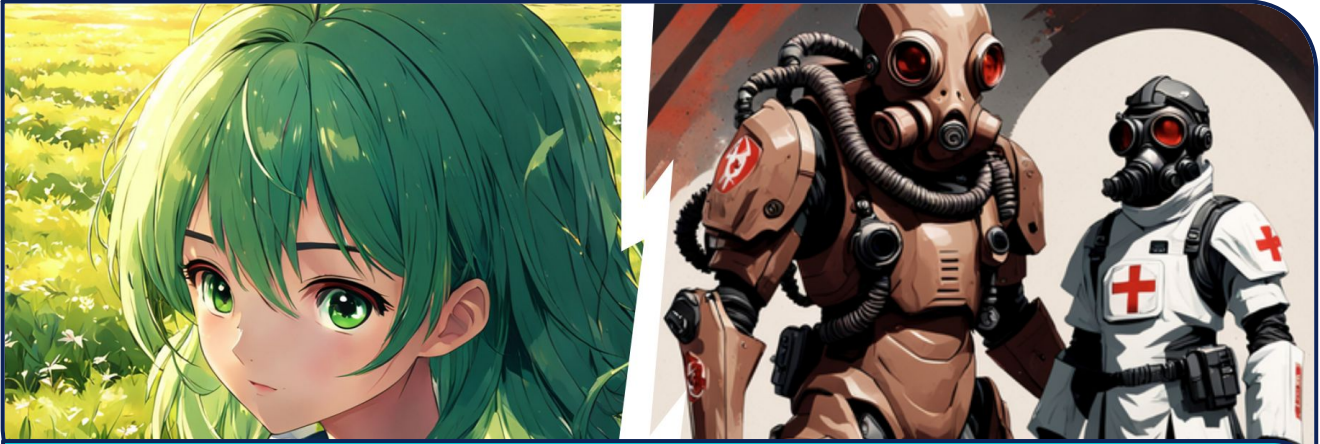
1. Après avoir reconnu le goût des raviolis qu'il a eu comme seul repas pendant 15 ans, Oh Dae-su (Choi Min-sik) affronte, lors d'un plan séquence en *side-scrolling* inspiré des *beat them all*, les hommes de main de Park Cheol-woong. *Old Boy* [올드보이] (2003) — Park Chan-wook.

EFFICIENCE OPÉRATIONNELLE & SUSTAINABLE



ii. Brownfield

ii. L'approche « Brownfield »



Greenfield vs Brownfield deux salles, deux ambiances...

Après plusieurs années d'exploitation, il est assez fréquent de se retrouver dans la situation où il est de plus en plus difficile d'ajouter ou de faire évoluer les fonctionnalités de son application ou système. En général, c'est à ce moment-là que se pose la question de réécrire complètement cet asset via un développement dit « *greenfield* ».

Ce n'est pourtant pas toujours un bon calcul car cet asset considéré comme legacy a de la valeur : c'est la représentation la plus tangible des règles métiers déployées en production. Une réécriture entraîne le risque de perdre une partie du métier et de déployer en trajectoire des **stratégies complexes de double run**.

L'approche dite « *brownfield* » — qui peut paraître peu ragoutante — prône une **reprise sous contrôle** du code existant de manière progressive ; elle est souvent **plus intéressante économiquement**. Elle permet une mise en exploitation plus rapide et s'inscrit dans une **démarche écoresponsable**.

Mais qu'est-ce que le Brownfield ?

Définitions

- Un code est dit **greenfield** quand le produit est naissant et n'a jamais connu de livraison en production.
- Un code est dit **brownfield** quand le produit est déjà en production, avec des dysfonctionnements et quelques anti-patterns.

Comme nous allons le voir, l'approche brownfield, c'est en quelque sorte une équipe qui joue à *Dr. House* et s'emploie à traiter un patient malade.



Pourquoi maintenant ?

L'approche **brownfield**, induisant des coûts constants et prévisibles, est **économiquement plus viable** :

- Elle évite de gérer deux instances en dual RUN et les coûts d'exécution afférents.
- Elle évite le risque de ne pas reprendre correctement la connaissance métier en production et un long tunnel de réécriture sans feedbacks.

Par ailleurs, avec le contexte d'**erooM**, l'approche **brownfield** porte une dimension **écoresponsable**.

Le brownfield porte une dimension écoresponsable

Non seulement nous allons devoir **optimiser** nos applications avec le contexte d'eroom, mais l'épuisement des ressources matérielles et l'augmentation du prix de l'énergie vont nous amener à favoriser des approches **brownfield** plus **écoresponsables**, afin de **recycler** et de reprendre le contrôle de nos assets existants. Car cela nous permet de conserver les mêmes technologies, les mêmes matériels, les mêmes langages de programmation, le même écosystème et de ne pas augmenter notre **empreinte carbone**.

Ainsi, en **cultivant vos produits brownfield**, vous engendrez des **économies, avec moins d'impacts pour la planète**.



La démarche scientifique que nous préconisons : diagnostic et traitement

Pour conduire un chantier brownfield, nous déployons sur le terrain une **Brownfield Squad** qui définit une stratégie de remise sous contrôle holistique en deux temps sur les aspects *métier, technique et organisationnel*.

i. L'équipe collecte les symptômes pour dresser un DIAGNOSTIC

- Elle collecte les douleurs : coût, T2M, lead time, change/fail rate, time to restore, perf., etc.
- Elle dresse un bilan de santé de l'application et de l'équipe : niveau de maîtrise fonctionnelle et technique de l'application ou du système.

ii. Puis elle prodigue un TRAITEMENT

- Elle adopte une approche **itérative** pour reprendre le code petit bout par petit bout et introduit ou améliore les pratiques méthodologiques **Software Craftsmanship**.
- Elle met en place des **indicateurs** qui permettent de monitorer l'état de santé du patient et contrôler que la situation ne se dégrade pas à nouveau. Les DORA metrics sont souvent pertinentes dans ce contexte.



Brownfield : une approche sociotech¹ pour la gestion du changement

Lors d'une démarche de remise sous contrôle d'une application ou d'un système, si l'approche *baby steps* et la mise en place de métriques de suivi sont clés, un des enjeux porte sur les aspects d'accompagnement du changement, principalement sur les pratiques de développement.

Nous présentons ci après, plusieurs pratiques issues du **Software Craftsmanship**.



« Lors d'une remise sous contrôle **brownfield**, la gestion du changement sur les aspects méthodologiques est déterminante : **pensez-vous que refaire quelque chose de la même manière aboutira à un résultat différent ?** »

Alexandre JEAMBRUN (feat. EINSTEIN)
CRAFT Practice Lead @ OCTO Technology

1. En 1952, Eric L. Trist invente le terme de « système socio-technique » pour souligner que les problèmes d'interaction des personnes avec les outils et les techniques ne résultent pas d'un hasard. Gilbert Simondon en 1958 défend l'idée de co-construction qui permet de poser autrement la question du rapport entre « société » et « technique » que par le déterminisme de l'un sur l'autre.

Le Software Craftsmanship à la rescousse avec trois types de pratiques

i. Exemples de pratiques de conception fonctionnelle

- ◉ Le **Domain-Driven Design** : concevoir un système découpé en sous domaines métiers indépendants, avec des responsabilités claires, et techno-agnostique.
- ◉ **L'EventStorming Big Picture**. L'objectif est de retrouver des narrations fonctionnelles. La qualité de l'information peut être imprécise mais permet d'obtenir un début de connaissance.
- ◉ Le **Model-Driven Design** est un mix de l'approche de modélisation au niveau du code qu'on appelle **Deep Modeling**, accompagnée de patterns techniques qu'on appelle **Supple Design**.
- ◉ Le **Domain Storytelling** est une approche orientée sur un flux fonctionnel qui permet d'être très explicite sur les intentions métier. Il existe des outils graphiques comme Domain Story Modeler, qui permettent de s'initier de manière ludique.



ii. Exemples de pratiques de conception applicative

- ◉ Les **Design Patterns** sont une collection de solutions techniques éprouvées permettant de structurer le code de manière durable.
- ◉ Le **Clean Code** est une série de principes visant à bonifier le code, comme les principes **SOLID** qui permettent d'apporter de la souplesse au code afin d'épouser de potentiels changements.
- ◉ Le **découpage du code en modules indépendants**. Nous devons briser les dépendances aux frontières fonctionnelles afin de forger un module métier qui deviendra un service spécifique. L'approche consiste à appliquer les patterns de l'ouvrage de Michael Feathers pour ne rien casser dans un premier temps.
- ◉ Le **Refactoring de design** est une collection de mouvements au niveau du code afin de corriger l'essentiel des **codes smells** — une collection d'anti-patterns qui permettent de poser un vocable commun sur des problèmes de design. L'objectif est d'améliorer la communication entre les développeurs et de favoriser la correction de code smells.



iii. Exemples de pratiques de développement

- ◉ **Mob programming** : pour améliorer la qualité du code, le partage de la connaissance et faire monter en compétence.
- ◉ Le **TDD** est la pratique la plus simple pour développer en mode design émergent au niveau d'une structure de code. Le **TDD outside/in** est une pratique avancée qui permet d'engendrer du code sur la base d'un test acceptance, où le code sous-jacent fera l'objet d'un développement avec le **TDD inside/out**.
- ◉ **Behavior Driven Development** est un moyen de transférer la connaissance des comportements métier vers l'équipe de développement, en explicitant les comportements fonctionnels attendus, tout en étant au plus proche du code.



Conclusion

En adoptant une nouvelle approche du développement applicatif qui intègre les principes du **Software Craftsmanship** de manière permanente et en reprenant itérativement le code existant (même celui qui semble difficile à maintenir, peu lisible, fortement couplé ou non testé), il est possible de rendre ce code à nouveau maîtrisable et agréable à utiliser pour l'équipe en charge.

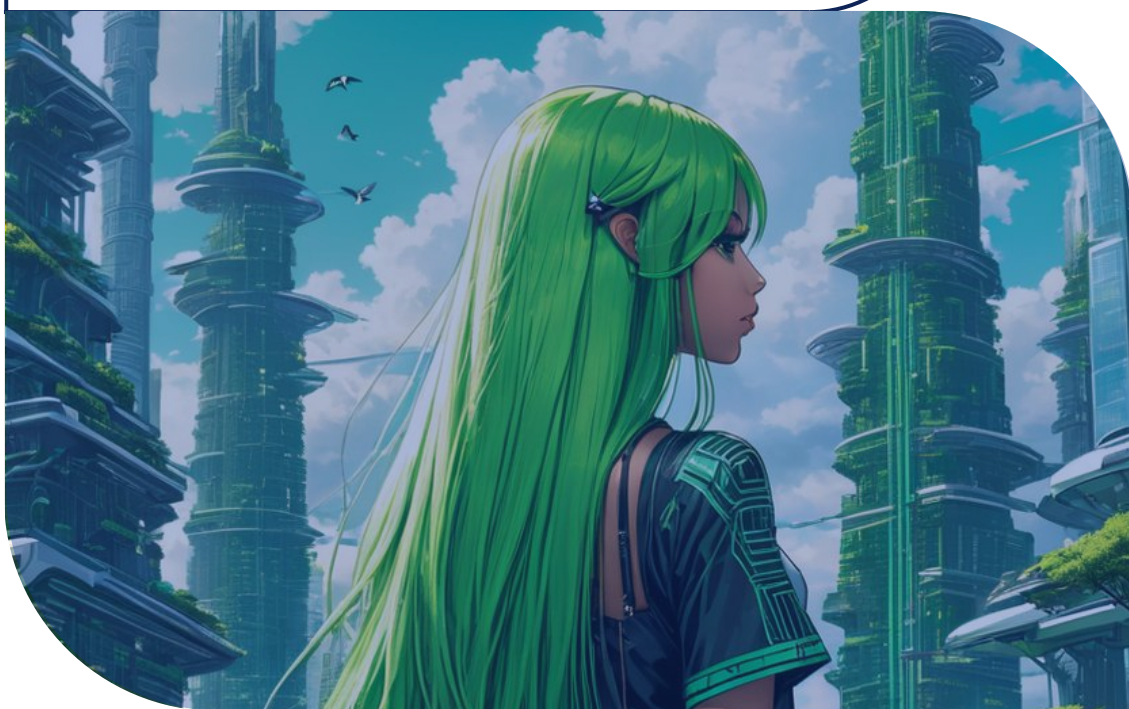
In fine, nos expériences révèlent que les pratiques, les patterns et les bonnes habitudes de développement sont bien plus importants que la technologie, le langage ou l'écosystème choisis. La démarche de reprise sous contrôle peut alors stimuler les équipes et devenir plus épanouissante que de bâtir une énième application greenfield.



Alors en 2025, comme Luke, vous aventurerez-vous dans les marécages de **Dagobah**¹ pour en découdre avec le **Brownfield** ou préférerez-vous les prairies **Greenfield** de **Naboo**² (qui ressemblent étrangement au Wallpaper Windows XP) ?

1. Dagobah est une planète tellurique, célèbre pour être le lieu où maître Yoda, dans son exil, forme Luke Skywalker à devenir un Chevalier Jedi. *The Empire Strikes Back* (1980) — George Lucas.
2. Naboo est une planète notamment célèbre pour être le monde d'origine de politiciens d'envergure galactique tels Sheev Palpatine et Padmé Amidala. *Star Wars: Episode I – The Phantom Menace* (1999) — George Lucas.

EFFICIENCE OPÉRATIONNELLE & SUSTAINABLE



iii. Accessibilité numérique

iii. Accessibilité numérique (a11y)



Il est temps de s'y mettre !

Pourquoi on en parle autant en 2024 ? Parce que la loi française se renforce pour suivre la directive européenne avec deux échéances majeures : dès juin 2025, tous les nouveaux produits et services numériques des entreprises privées devront être accessibles, tout comme les services publics. Et les parcs existants ? Des décrets imposent la mise en conformité de tous vos parcs existants d'ici 2030. Oui, TOUS.

Les contrôles ont déjà commencé, avec des amendes salées à la clé. Investir maintenant, c'est aussi limiter les coûts à venir. Il est donc temps de considérer les opportunités offertes par l'accessibilité !

Opportunités de l'accessibilité numérique

\$6,9

milliards

perdus par le
e-commerce
par manque
d'accessibilité.



20 à 40%

des internautes ont un
accès difficile, partiel
ou impossible aux
informations et
services en ligne.

+300%

explosion de plaintes
pour non accessibilité
en 2020 aux US (et ça
continue de croître) !

Bon nombre d'organisations réalisent qu'intégrer l'accessibilité amène de nombreux bienfaits en limitant les risques légaux, en renforçant la présence de la marque, en améliorant l'expérience utilisateur autant que la productivité interne.

— Paul Smyth, Barclays.

Pourquoi maintenant ?

L'accessibilité numérique n'est pas un sujet nouveau mais la contrainte légale s'accroît en France, afin d'appliquer la directive européenne sur l'accessibilité (EAA) qui s'impose à tous les États membres depuis 2019.



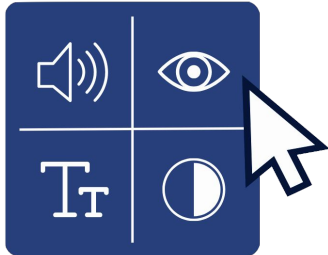
En effet, de nouveaux textes de loi renforcent les échéances et sanctions, augmentant le **risque judiciaire**. Après le secteur public, deux échéances majeures sont à retenir pour les entreprises privées :

- **à partir du 28 juin 2025**, les *nouveaux* produits et services devront être **accessibles dès leur entrée en service** ;
- **au 28 juin 2030, TOUS les produits et services numériques** vendus ou utilisés au sein de l'UE devront être accessibles, qu'il s'agisse de site internet, intranet, extranet, B2C ou B2B, app mobile, progiciel, mobilier urbain numérique, billetterie de transport, livre numérique, etc.

Les sanctions s'accroissent, avec actuellement **deux amendes**, cumulatives, à hauteur de 50k€ + 25k€, par produit et par semestre. Certains services risquent de plus des **contraventions de 5e catégorie** et des astreintes journalières (plafonnées à 300k€). Sachez que les autorités de contrôle ont été désignées et que **les contrôles ont déjà commencé** pour les services publics. La première condamnation pour inaccessibilité numérique a été prononcée le 6 juin 2024.

Plus de choix : l'accessibilité numérique, il va falloir s'y mettre !

Fuyez l'arnaque au bouton magique



Plusieurs éditeurs vendent des « *solutions d'accessibilité* » sous la forme de surcouches¹.

Celles-ci proposent de rendre accessible les contenus existants en ajoutant des fonctionnalités d'adaptation (agrandissement des caractères, renforcement des contrastes, etc.). Certaines d'entre elles proposent même de réparer automatiquement les problèmes d'accessibilité.

La promesse est alléchante : mise en place rapide, sans modification à effectuer dans le code...

Mauvaise nouvelle : ces solutions sont en réalité inefficaces. Et elles ne protègent pas du risque légal.

L'ajout de fonctionnalités n'est PAS ce qui rend un produit accessible.

Souvent même, ces fonctionnalités ajoutent des problèmes d'accessibilité !

Voir le site dédié : overlayfactsheet.com

1. en anglais « accessibility overlays ».

Ne commencez pas par un audit RGAA

Étant donné qu'il existe une norme d'accessibilité, la tentation est forte de s'y mesurer, en commençant par « *un audit pour savoir où on en est* ».

Mais le RGAA n'est qu'une grille de contrôle final, inadaptée pour un état des lieux initial — et qui ne dit pas comment faire. **Si aucune action particulière n'a été entreprise, l'audit ne vous apprendra rien.**

En effet, l'accessibilité n'est pas une bonne surprise qui se découvrirait au hasard d'un contrôle, mais le résultat d'une démarche volontaire incluant des **gestes de fabrication précis**.

In fine, en l'absence de démarche, **l'inaccessibilité est certaine** : pas besoin d'audit pour le découvrir.



Commencer par un audit RGAA et corriger point par point ? Autant gravir le volcan qui domine le nord-ouest du Mordor¹.

1. Le Mordor (« pays noir » en sindarin) est une région de la Terre du Milieu, univers de fiction créé par J. R. R. Tolkien. Dans ce troisième tome du Seigneur des anneaux, Le Retour du roi, Frodon et Sam doivent en traverser les étendues arides pour accomplir leur mission : détruire l'Anneau unique en le jetant dans les flammes de l'Orodruin, le volcan qui domine le nord-ouest du Mordor.

Pire, un audit à ce stade peut s'avérer contre-performant : la quantité d'erreurs soudainement révélées, même si non exhaustive, peut décourager durablement les équipes à l'œuvre.

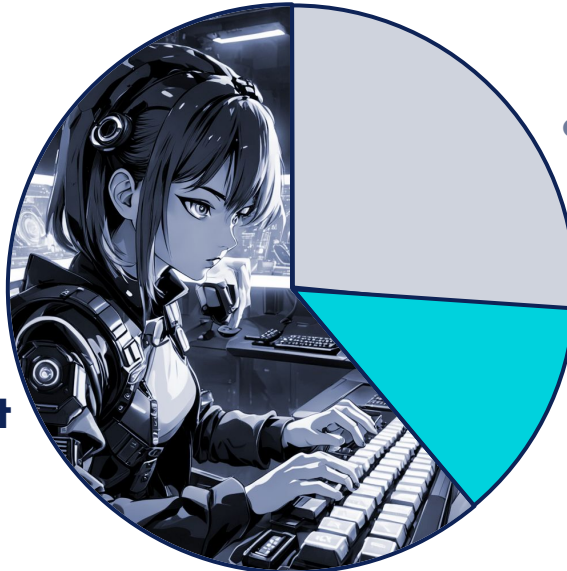
À plus forte raison, **l'approche corrective**, hélas si courante, à coup d'audits seulement, **est douloureuse pour les équipes, peu pérenne et onéreuse**. Il existe d'autres façons d'évaluer, plus simples et progressives.

Exemple de rapport d'audit RGAA : plus de 60 pages Word et des milliers de lignes Excel...



Engagez un travail d'équipe

61 %
dev
web front



26 %
contenus

13 %
design
(UX+UI)

Considérer les critères du RGAA selon les compétences nécessaires à leur réussite révèle une forte proportion technique :

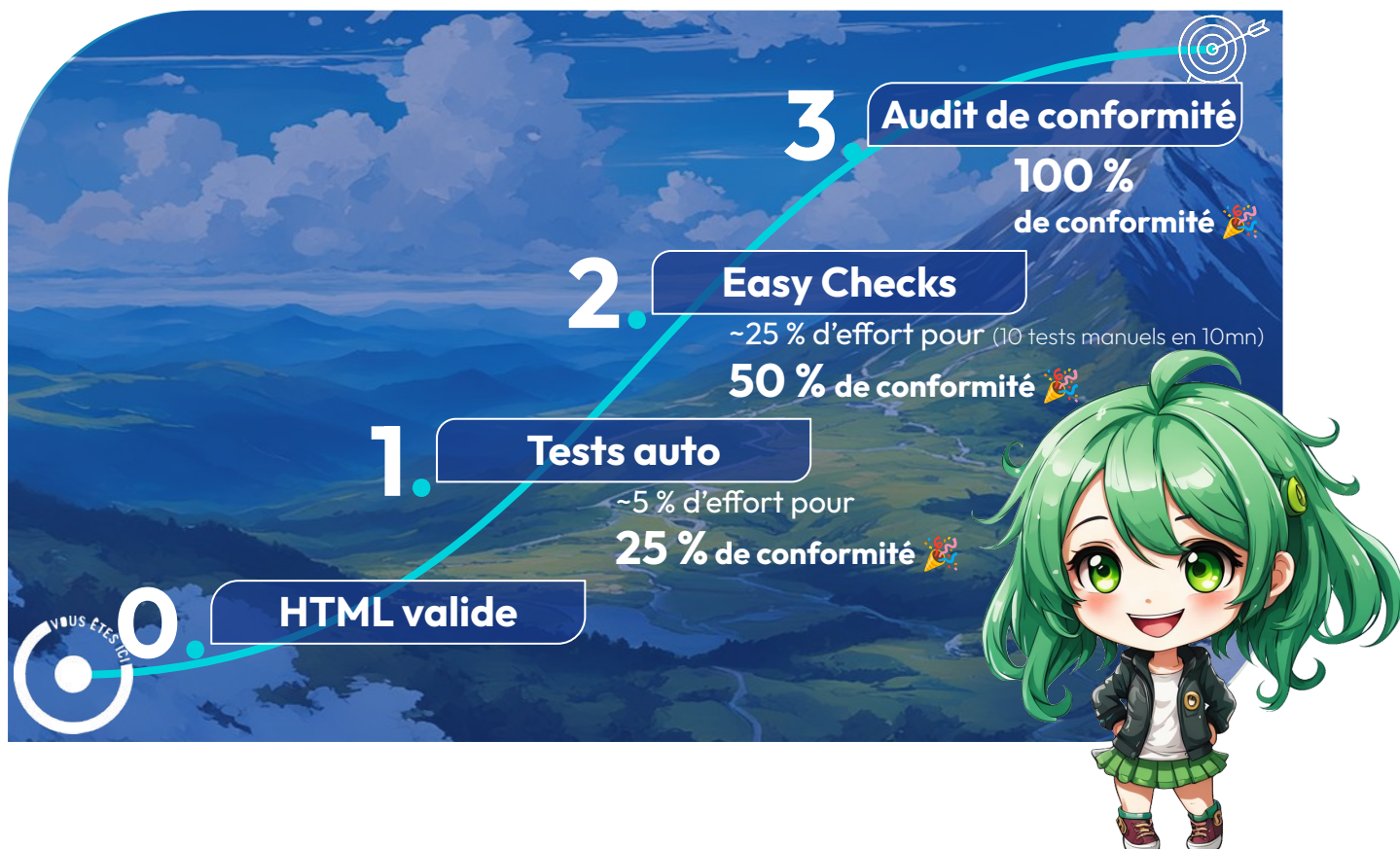
- contrairement à l'idée reçue, l'impact du design (UX+UI) sur l'accessibilité est plutôt mineur (à peine 13%) ;
- le rôle du **développement web front**, en particulier des langages HTML, JavaScript et ARIA, est quant à lui déterminant (à hauteur de 61%) : c'est un savoir-faire technique à acquérir ;
- enfin, un bon quart des critères concerne les contenus : textes, images, graphiques, PDF, vidéos, transcripts, etc.

Il faut surtout en retenir que tous les profils à l'œuvre sont concernés : la réussite de l'accessibilité est un **travail d'équipe** sous la houlette des PO.

Déployez une démarche progressive

Que ce soit pour les nouveaux produits ou pour ceux du parc existant, nous vous recommandons d'adopter une **démarche progressive**, que nous avons éprouvée avec succès, pour gravir — sans efforts — la montagne de l'accessibilité.

0. **La validité du code HTML est un prérequis** : adoptez le [HTML validator](#) pour éviter les erreurs de codage.
1. **Automatiser les tests d'accessibilité** : déployez des outils dédiés (comme [Wave](#) ou [Axe](#)) dans votre CI/CD, pour stabiliser ~25% de conformité.
2. **Inspection manuelle** ([Easy Checks W3C](#)) : 10 tests manuels en 10 minutes à inclure dans vos recettes, pour ambitionner 50% de conformité.
3. Enfin prêt pour l'**audit de conformité** !



Conclusion

Lancez dès à présent une démarche d'accessibilité numérique. En effet, de nouveaux décrets et organismes de contrôles imposent des sanctions en cas de non conformité :

- ◉ à partir du 28 juin **2025** pour les **nouveaux produits** ;
- ◉ au 28 juin **2030** pour votre **parc existant**.

Ne foncez pas sur l'audit de conformité : adoptez une **démarche progressive** en quatre étapes.



Alors en 2025, comme **Leeloo**¹,
offrirez-vous un **multipass** pour **accéder**
sans peine à vos **services numériques** ?

1. Leeloo (Milla Jovovich) est l'être parfait qui doit sauver la planète. Tout est dit. *Le Cinquième Élément* (1997) — Luc Besson.

EFFICIENCE OPÉRATIONNELLE & SUSTAINABLE



iv. Développement hybride

iv. Développement hybride

Définition

Le **développement hybride**, ou bien le **développement multi-plateformes**, consiste à utiliser un framework de développement qui permet de créer des applications pour plusieurs plateformes, à partir d'une **base de code unique**.

Cette approche cible en général les plateformes *iOS* et *Android*, voire *Web*.



Pourquoi maintenant ?

Dans un contexte nécessitant une **efficacité opérationnelle** et où l'on observe une **stabilisation des écosystèmes techniques**, la promesse du développement hybride est alléchante.

En effet, l'hybridation permet de **rationaliser les filières de développement** mobile, voire Web, et donc d'**optimiser vos coûts** en les divisant par deux ou par trois, tout en consommant **moins de ressources** logicielles et matérielles !

Montée en puissance de l'hybride natif avec deux framework stars qui s'affrontent



React Native & Flutter



Deux frameworks portés par deux géants du Web sortent du lot :

- ◉ **React Native** porté par *Facebook*, qui l'utilise dans plusieurs de ses apps.
- ◉ **Flutter** porté par *Google*, qui propose un large éventail de plugins.

On estime¹ aujourd'hui que 20% des applications sur les stores sont réalisées en *Flutter* et en *React Native*, avec en exemple des superstars des stores comme les applications *Skype*, *Google Pay* et *Amazon* qui sont réalisées en cross-platform.

1. Selon les sources appbrain et Google.

Bien plus que le choix d'une stack mobile

Le choix d'une stack mobile n'est pas à prendre à la légère, car c'est en réalité plutôt un choix de **stratégie des filières front-end**, avec des impacts **techniques** certes, mais aussi **organisationnels** et **financiers**.

Pour simplifier la problématique, le choix d'une stack mobile, c'est un peu comme choisir un **shampooing** : on peut acheter indépendamment son shampooing, son après-shampooing, et son gel douche ! Mais on a aussi les options « **2 en 1** » voire « **3 en 1** ».



Dans ce contexte, pour choisir votre stratégie de filières front-end, notre préconisation est de considérer **3 scénarios** (que nous détaillons sur la page suivante).

Sachant que les deux premiers permettent une **rationalisation des filières front-end via du développement multi-plateformes**.

1. Scénario « 3 en 1 » — PWA

Un site web responsive ou PWA sera parfois suffisant si :

- On ne cible pas un **usage récurrent** avec des utilisateurs qui reviennent très régulièrement (qui justifie l'installation d'une app).
- On ne cherche pas à bénéficier de tout le support de l'OS ou des **performances du device**.
- On est prêt à faire des compromis concernant la présence sur les **stores**. Gardons en tête que le processus de validation des PWA est complexe et évolue, sachant qu'Apple tente régulièrement de bloquer l'usage des PWA sur son store en raison des achats in-app. Il faut donc être prêt à pouvoir se couper d'une partie de l'audience.

Retenons un point de vigilance : Pour avoir une belle « *expérience native* » avec une PWA, des développeurs très chevronnés web et mobiles sont nécessaires.



Dans ce scénario, on a potentiellement une seule filière de développement : **Web** qui permet également d'adresser les smartphones **iOS** et **Android**.

2. Scénario « 2 en 1 » — Hybride natif

L'**hybride natif** est un **excellent choix** (Sweet spot) si le Web n'est pas suffisant, et c'est **notre recommandation par défaut** pour **offrir une belle expérience mobile**. Il permet un bon compromis entre qualité et performance des apps / coûts et délais de mise en œuvre, et n'a pas de contraintes pour le déploiement sur les stores.



Dans ce scénario, on a potentiellement deux filières de développement : **Web** et **hybride mobile**.

3. Scénario « Spécifique » — 2 apps natives

À privilégier pour offrir la **meilleure expérience mobile** :

- pour des cas d'usage gourmands, qui nécessitent toute la puissance du smartphone (ex: 3D, AR/VR) ;
- pour profiter des dernières nouveautés des OS ou SDK spécifiques qui ne sont pas compatibles avec une approche hybride ;
- si vous avez un budget confortable et un accès facile à des populations de développeurs iOS et Android.



Dans ce scénario, on a potentiellement trois filières de développement : **Web**, **iOS** et **Android**.



1. 3 in 1 PWA

Choix par défaut

☆☆☆ XP mobile
☆☆☆ Store
☆☆☆ Coûts / T2M

2. 2 in 1 Flutter Hybrid app



Sweet spot pour une belle expérience mobile

☆☆☆ XP mobile
☆☆☆ Store
☆☆☆ Coûts / T2M

3. 2 apps natives



Pour une expérience mobile premium

☆☆☆ XP mobile
☆☆☆ Store
☆☆☆ Coûts / T2M

L'hybridation permet de **rationaliser** les filières de *développement mobile*, voire *Web* !

En 2025, pour votre **stratégie de filières front-end**, vous aurez trois solutions :

1. Un **site web responsive** ou **PWA**, souvent suffisant.
2. **L'hybride natif**, un **excellent choix** si le Web n'est pas suffisant : c'est **LA solution** à considérer en 2025 pour **offrir une belle expérience mobile**.
3. Les apps natives si vous n'avez **pas de contraintes budgétaires fortes** et que vous souhaitez une **expérience mobile premium**.



Alors en 2025, les **applications hybrides** deviendront-elles un **standard de fait** ? Ou bien de redoutables **hybrides humain/xénomorphe**¹ comme le **Newborn d'Alien 4**, dont Ellen Ripley devra nous débarrasser ?

1. Le Newborn est un être hybride entre humain et xénomorph enfanté par une reine (Ripley 8) qui a été clonée depuis lieutenant de première classe Ellen Ripley (Sigourney Weaver). *Alien Resurrection* (1997) — Jean-Pierre Jeunet.



04

ORGANISATION.

- i. Software engineering distribué
- ii. T-shaped skills
- iii. Career path Tech



LES CONTRIBUTEURS DE CE CHAPITRE



**Antoine
CHANTALOU**

Head of Software Engineering
antoine.chantalou@octo.com



Armen OZCELIK

API Lead
armen.ozcelik@octo.com



**Marie
HERNANDEZ**

Software Engineering HR Lead
marie.s.hernandez@octo.com



**Marie-Agnès
GUY**

CEO Chief of Staff
marie-agnes.guy@octo.com



04. ORGANISATION



Des organisations **distribuées** **performantes et scalables**

Dans un contexte d'**archipélisation des lieux de travail** suite au Covid-19 et avec situation socio-économique sous tension qui induit une **pression sur les prix**, nous constatons **3 Trends sur le plan organisationnel** :

1. L'essor du **software engineering distribué**
2. Une séniorisation des équipes avec des profils **T-shaped skills**
3. Une émergence d'un **career path TECH** pour être attractif et retenir ces profils recherchés

ORGANISATION



i. Software engineering distribué

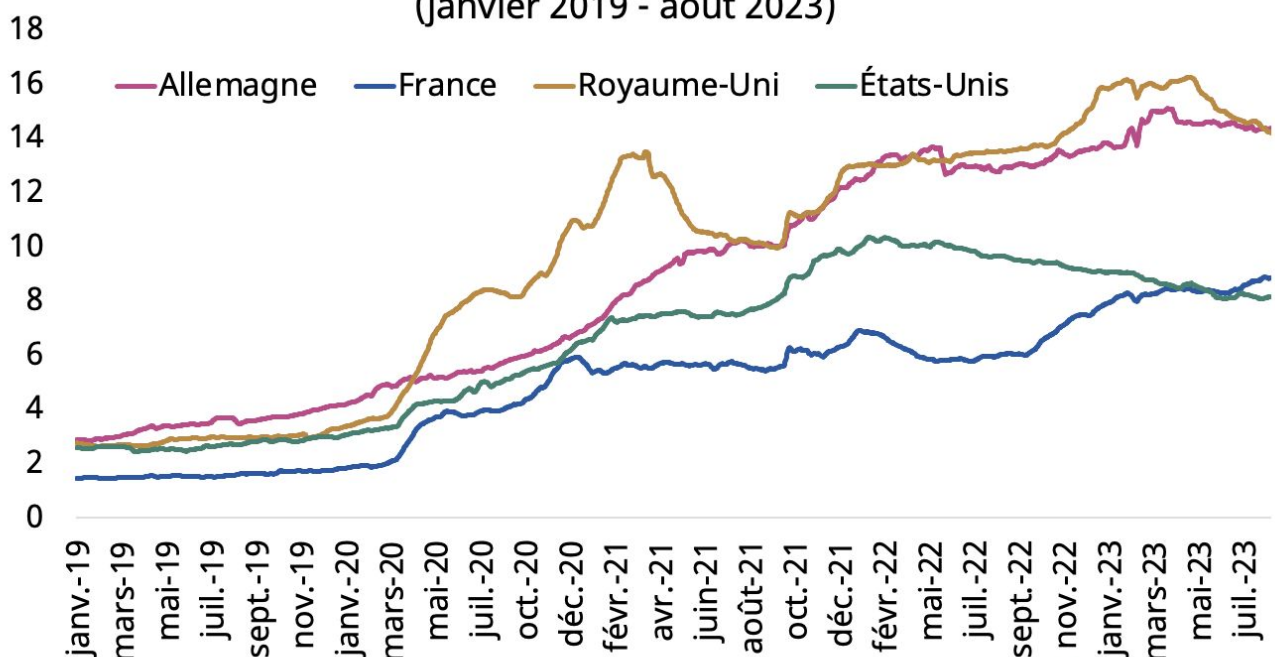
i. Software engineering distribué

Un changement de paradigme du travail dans la Tech

Depuis la pandémie Covid, nous sommes confrontés à une archipélisation des lieux de travail. En France, selon les statistiques de l'INSEE, le télétravail a connu une forte progression : en 2023, environ 47% des entreprises françaises y avaient recours. On constate cependant une baisse visible de la part des salariés français travaillant à distance par rapport au pic de 2020, probablement liée au fait que de nombreuses entreprises ont appelé à en finir avec le télétravail. Il demeure néanmoins très au-dessus des niveaux pré-pandémiques et est toujours en croissance en Europe, selon l'outil Remote & Hybrid Job Tracker d'Indeed.

Le télétravail toujours très au-dessus des niveaux pré-pandémiques

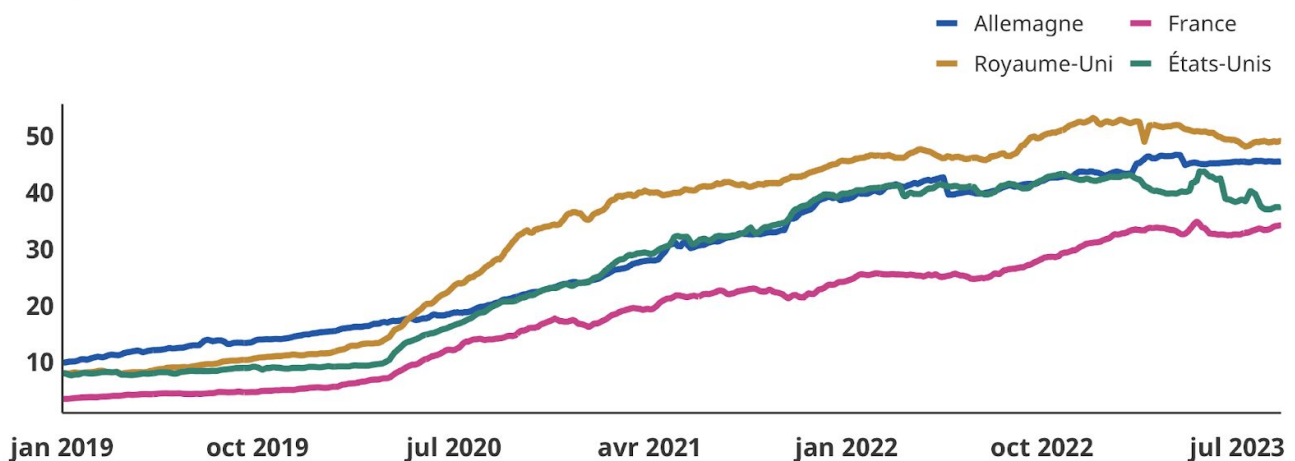
Proportion des annonces hybrides ou en télétravail sur Indeed
(janvier 2019 - août 2023)



Si le **ratio de télétravail dans la Tech** est en recul aux États-Unis, ce qui est probablement lié au fait que Twitter, Google et Facebook ont changé leur fusil d'épaule en 2021, **il continue d'augmenter en France** et est par ailleurs bien plus haut que dans les autres secteurs.

Le télétravail en recul dans la "Tech" aux États-Unis

Proportion des annonces hybrides ou en télétravail sur Indeed
(janvier 2019 - août 2023)



Source : Indeed

indeed

En effet, certains métiers se prêtent plus que d'autres au télétravail. Pour la **Tech**, et **principalement les software engineers** qui n'ont besoin que d'un laptop bien musclé et d'une connexion internet qui l'est tout autant, c'est à priori un mode de travail adapté. D'autant que les entreprises — depuis que l'on clame "Software is eating the World" — s'efforcent généralement de **retenir les développeurs**, souvent hautement qualifiés et leur **offrent donc des modalités de travail flexibles**.

En France, les offres proposées aux développeurs, avec du travail à distance ou hybride, dépassent les 34%. Un chiffre bien supérieur aux 8,9% de travail à distance ou hybride proposés sur l'ensemble des offres d'emploi !

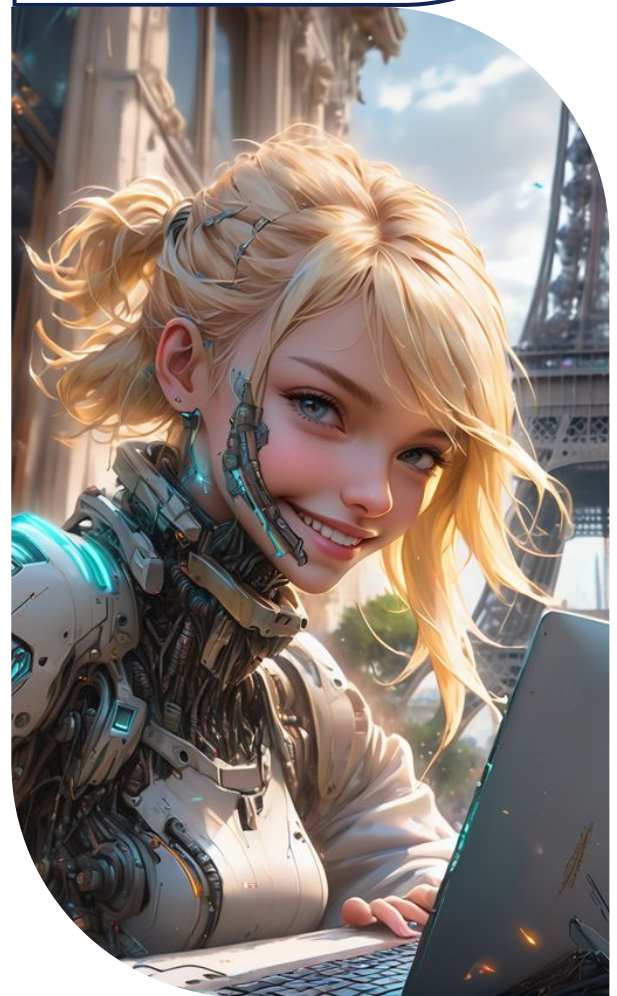
La hype du développeur freelance

D'après une étude menée par BCG et la plateforme de freelancing Malt, ce sont les métiers de la technologie qui regroupent la majeure partie des freelances : en France, 29% des freelances travaillent dans l'industrie de la technologie et 59% d'entre eux travaillent en *remote*.

Et selon la plateforme Fiverr Workspace, « The cash is in the code » : les freelances Tech sont les mieux rémunérés en télétravail.

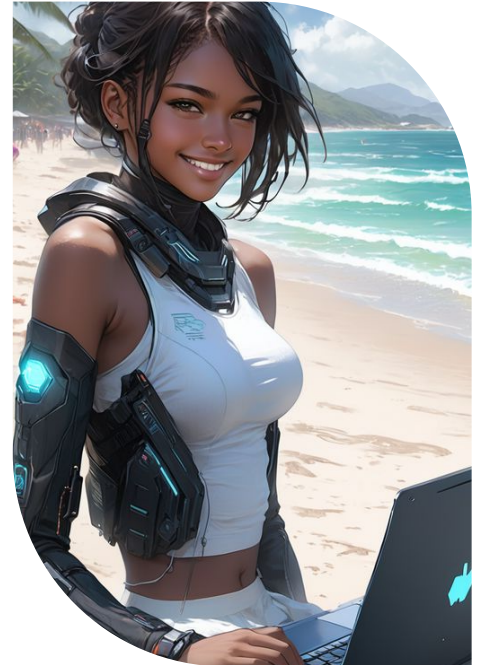
Ainsi, la liberté de pouvoir travailler en remote combinée à des salaires élevés, expliquent pourquoi une part des Millennials, cette génération de digital natives née entre 1980 et 1995, s'est lancée dans une activité de freelancing dans la tech.

Selon Fiverr, 80% des interrogés souhaitent continuer à travailler en remote indéfiniment (55% sont en full remote et le reste en hybride). 43% sont en freelance.



Les digital nomades : l'autonomie du freelancing et pouvoir travailler en voyageant

Le monde professionnel, qui connaît depuis la crise sanitaire un bouleversement de ses habitudes, va très probablement continuer de se transformer dans les prochaines années. Avec l'arrivée sur le marché du travail de la génération Z, née entre 1996 et 2010, débarque une **vision 2.0 du monde**, des aspirations **d'autonomie** et de changement régulier d'activité, une **quête de sens**.



Cette population se retrouve pleinement dans le concept du freelancing, voire du **digital nomadisme**. Ces jeunes pluri-tâches pour qui « *Slack is the new office* » risquent d'être davantage attirés par les **missions à sens** que par les **fonctions à responsabilités**.

C'est ce que révèle une étude menée par Fiverr, qui indique que 72% des Français de la génération Z envisagent de travailler en tant que freelance, avec pour objectif un bien-être financier (48 %) et la capacité à **voyager pour le travail** ou à **travailler d'où ils le souhaitent** (32 %).

Les avantages du software engineering distribué

Chez OCTO Technology, dans le cadre des missions que nous réalisons pour nos clients, nous avons vécu ces tendances — comme vous probablement — avec une augmentation du ratio de télétravail dans la Tech.

Ces quatres dernières années, nous avons connu une majorité de missions chez nos clients où **l'hybride** est devenu le **mode de travail par défaut**, avec un équilibre oscille entre 2 et 3 jours de télétravail par semaine. Mais ce ratio dépend des secteurs d'activité, de la culture d'entreprise et du type de mission.

Nous avons identifié plusieurs **avantages** :

- ◉ **Moins de perte de temps** liés aux aspects logistiques (principalement pour les déplacements).
- ◉ La possibilité de **mieux organiser son temps de travail** et de l'articuler avec son temps personnel.
- ◉ Un bien-être qui en définitive peut **rendre certains employés heureux** et qui, en boomerang, **s'investissent pleinement sur leurs missions**.
- ◉ Une **sacralisation des temps de regroupement**.
- ◉ Une **amélioration de la productivité pour les employés seniors organisés**.
- ◉ Une **réactivité accrue** pour certaines interventions en production sur des horaires non standards.
- ◉ Potentiellement, la capacité de **recruter des profils Tech experts** sur un **territoire plus étendu**.

Les douleurs du software engineering distribué

Nous avons collecté des douleurs qui sont aujourd'hui bien connues :

- La **complexité de communiquer** quand on n'a plus tous les membres de l'équipe sous la main au même endroit : oui il y a *Slack*, mais avec un léger asynchronisme. Il y a donc un **effort de synchronisation très important à fournir pour aligner les membres de l'équipe**. Voire l'apparition de tensions plus fréquentes causées par des incompréhensions venant des discussions par écrit.
- La **perte des moments informels** et un **détachement** peuvent s'opérer, sans la bière du jeudi et les croissants du vendredi, et dans certains cas un **manque d'engagement** pour l'équipe : comme on dit, « *Loin des yeux, loin du cœur* »¹. Il peut aussi arriver qu'un développeur adopte une posture plus « *exécutante* », a.k.a « *je prends mes tickets et voilà* ». Tout cela peut même aboutir à une forme d'isolement social.
- Le télétravail est un **frein au démarrage d'une mission**, pour lancer un produit, car l'équipe ne se connaît pas forcément et a besoin de « *se renifler* ».
- Des **difficultés à déceler rapidement la perte de régime** (fatigue, stress, démobilisation) chez la population en télétravail. Là où on les lisait dès le lundi matin sur la tête des gens.
- Une **overdose du virtuel** : sauter de *Meet* en *Zoom* et passer sa journée en visio avec finalement peu de pauses.
- Une **monté en compétence des juniors** qui peut être **ralentie**, car le présentiel facilite les interruptions informelles — qui peuvent être des vecteurs de transmission de la connaissance.

1. « *Quantum oculis, animo tum procul ibit amor* » dans sa version originelle.

Software engineering distribué & offshore : même combat ?

Dans ce contexte où le télétravail se généralise, le sujet de **l'offshorisation** a été remis sur le devant de la scène. On entend souvent : « *puisque une partie des membres de l'équipe n'est pas présente de temps à autre, alors pourquoi ne pas tout offshoriser ?* ». Effectivement, télétravail et offshore peuvent parfois se chevaucher, mais ces deux modèles, s'ils présentent quelques similitudes, ont en réalité des différences majeures.

Les similitudes

1. L'offshore et le télétravail permettent d'accéder à un **vaste bassin de talents**, sans se limiter à une **zone géographique** précise.
2. Les deux approches permettent des **économies en termes de coûts** (locaux, variation des salaires en fonction des régions ou pays, etc.).
3. Les deux approches induisent une **complexification de la gestion du delivery des produits**, avec des points d'attention sur l'engagement des employés qui peut s'amoinrir avec la distance, la diffusion de la culture d'entreprise et la coordination des équipes.

Les différences

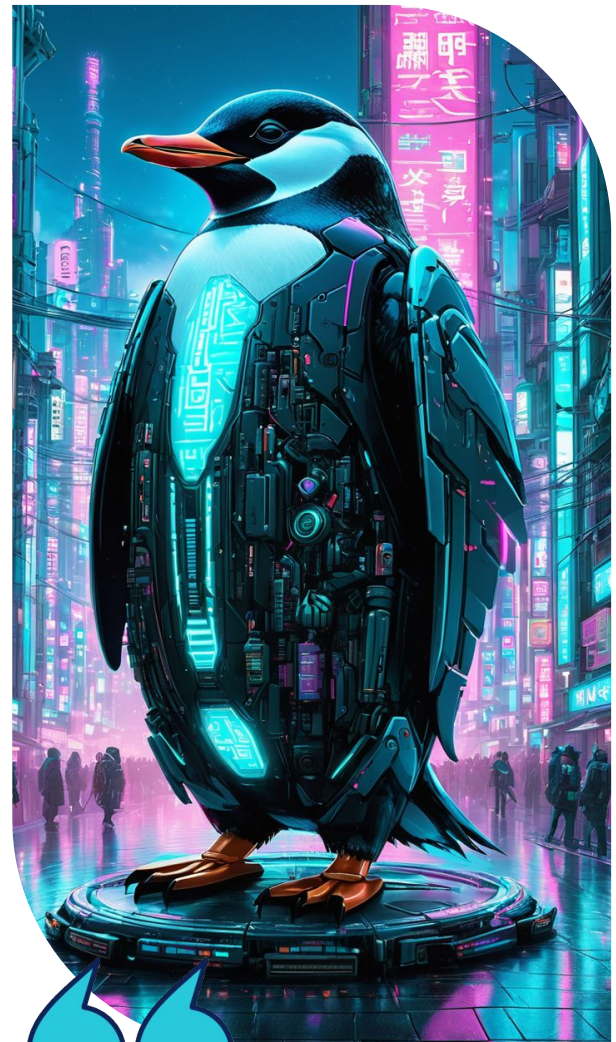
1. La **distance dans le temps et l'espace** : une stratégie d'offshorisation est souvent motivée principalement par la **volonté de réduire les coûts** (c'est le driver principal pour 57% des cas selon une étude réalisée par Deloitte), et donc de déporter les **tâches de développement** dans des **pays éloignés**. La réduction des coûts peut alors être plus importante, mais la barrière linguistique et culturelle, ainsi que les problèmes de fuseaux horaires, peuvent devenir des obstacles. Un modèle de delivery qui use du télétravail n'induit pas nécessairement une distance aussi lointaine, et bien souvent, les membres de l'équipe peuvent se retrouver en présentiel.
2. La **topologie des équipes est souvent bien différente** : dans un modèle offshore classique où les développeurs sont outsourcés, on introduit un découpage **horizontal** de l'équipe. Des **silos sur le principe client/serveur** peuvent se créer entre l'équipe onshore (souvent les acteurs métier et le pilotage) et l'équipe offshore (souvent les développeurs qui exécutent). Alors que dans le modèle de delivery qui use du télétravail à un certain ratio, avec des jours de rencontre synchronisés, les membres de l'équipes travaillent ensemble, de manière **verticale**, souvent en *SQUAD* (c'est-à-dire avec un même but autour d'un *produit*, une *feature* ou un *composant*) sans les travers d'une organisation horizontale et les problématiques de dilution de la responsabilité, d'attentes mutuelles entre équipes et de « *renvoi de la balle* » en cas de dysfonctionnement.

Le software engineering distribué, plus proche des pratiques de développement open-source

Il est assez inexact de confondre les deux initiatives **télétravail** et **offshore**, dans la mesure où les équipes sont alignées autour d'un produit pour le télétravail et où cette approche n'induit pas un outsourcing d'une partie des activités (le développement) avec la création de silos et la complexité organisationnelle que cela peut générer.

L'approche remote est plus proche du processus de réalisation des **projets open source**, où les acteurs travaillent en **distanciel** de **manière collaborative**.

Et lorsque l'on parle d'*open-source* et de développement remote, et que l'on se demande si cela fonctionne depuis son explosion dans les années 90 ? Avec l'exemple de *Linux* (l'OS déployé sur ~96% des top-serveurs) ou de VLC en France, etc. la réponse est **OUI**.



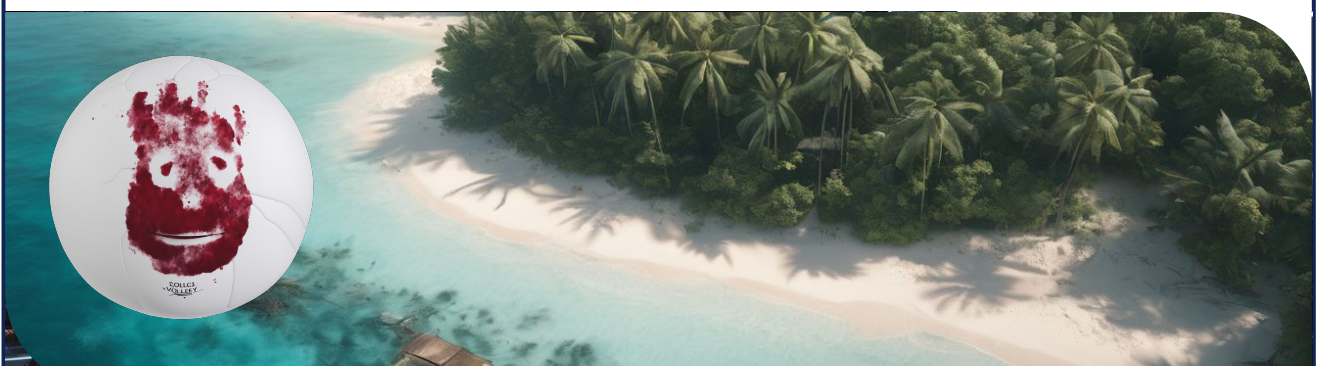
“96.3% of The top 1,000,000 web servers use Linux. (ZDNet)”

“85% of smartphones are powered by Linux (Hayden James).”

Il est difficile de prédire comment se conclura l'épopée du télétravail post-covid. La Gen Z Tech, avec ses aspirations d'autonomie, en quête de sens, est demandeuse de télétravail, voire de freelancing ou de digital nomadisme. Et de l'autre côté des cordes, de nombreux dirigeants, Boomers ou Gen X, sont nostalgiques d'une époque Caméra Café et militent plutôt pour un retour des troupes. Des modèles intermédiaires pourraient émerger ou aboutir à un consensus :

- Favoriser le télétravail pour les **populations méritantes**.
- Déployer un mode **hybride** en s'accorder sur des jours identifiés où l'ensemble de l'équipe est présente sur site, avec une liberté sur le reste des jours.
- Proposer **plusieurs modes** (full-présentiel, hybride, full-remote) en fonction des métiers.

Quoi qu'il en soit, que nous en soyons promoteurs ou détracteurs, en 2025, nous serons confronté au **Software Engineering distribué**.



Alors en 2025, vous isolerez-vous comme **Chuck**¹ dans **Cast away** avec votre laptop et **Wilson** ? Ou bien pencherez-vous pour l'option **Michael Scott** dans **The Office** ?

1. Chuck Noland (Tom Hanks), travaillant pour la compagnie de fret aérien FedEx, est appelé en urgence sur un vol le soir de Noël, alors qu'il dînait avec sa fiancée et sa famille. Victime d'une grave avarie en traversant une tempête, l'avion-cargo s'écrase dans l'océan Pacifique. Il se retrouve échoué sur une île déserte. *Cast Away* (2000) — Robert Zemeckis.

ORGANISATION



ii. T-shaped skills

ii. T-shaped skills

Définition

Les profils « **T-shaped skills** » sont des personnes avec une **dominante d'expertise** très forte et très profonde (le pied du T) et une **culture générale** sur un grand nombre de sujets connexes (la barre du T). Ces profils sont essentiels dans une équipe car ils peuvent monter en compétences rapidement et contribuer efficacement dans des écosystèmes techniques qui évoluent en continu.



Pourquoi maintenant ?

Toujours dans un **contexte socio-économique sous tension** qui nécessite une **efficacité opérationnelle accrue**, nous avons vu plusieurs équipes de software engineering se recentrer sur des profils plus **séniors** et plus **T-shaped**.

En effet, il n'est pas envisageable d'empiler à l'infini, au sein d'une SQUAD, des profils experts dès qu'un besoin métier apparaît et tire de nouvelles solutions technologiques.

« T-shaped skills » & séniorité



Le développement de profils « **T-shaped skills** » va souvent de pair avec la **séniorité**, car l'expérience démontre que les **pratiques**, les **patterns** et les bonnes habitudes de développement sont bien plus importants que la **technologie**, le **langage** ou l'**écosystème sous-jacent**.

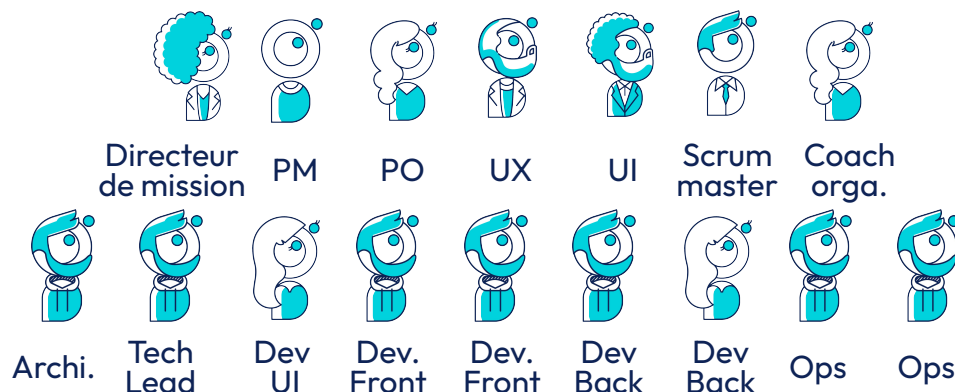
Par ailleurs, la **séniorisation des développeurs** est aussi une dynamique amplifiée par l'arrivée des assistants de code GenAI qui requièrent une maîtrise des pratiques Software Craftsmanship.

(Voir notre [section](#) sur ce sujet).

Nos retours d'expérience

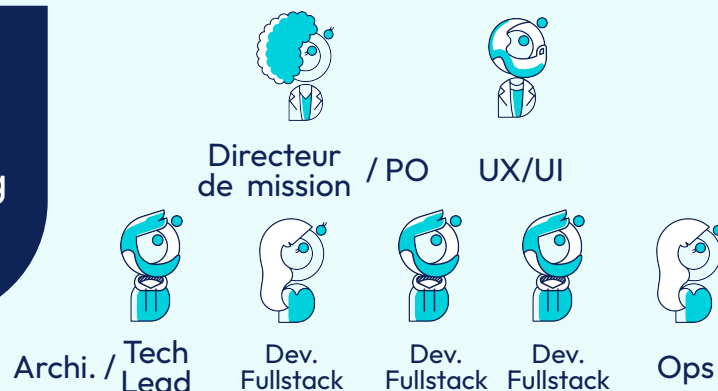
Si nous constatons que la trend du « *T-shaping* » s'est développée sur les populations de software engineers, c'est aussi une tendance que nous avons observé sur des profils moins tech des équipes de delivery. En 2023 et 2024 ont été privilégiés, dans certaines équipes, des profils multi-casquettes plutôt qu'un cumul des profils¹ : *PM*, *PO*, *UX*, *UI*, *Scrum master*, principalement pour des raisons de coûts et d'efficacité opérationnelle. (Voir illustration ci-dessous).

Équipe junior avec des expertises ciblées



Équipe senior avec des profils T-shaped skills

Profils seniors plus onéreux, mais moins d'ETP², pas besoin de coaching et moins de synchronisation.



1. PM (Product Manager), PO (Product Owner), UX (User eXperience), UI (User Interface).

2. ETP : Équivalent temps plein.

Conclusion

Les profils « **T-shaped skills** » sont des experts, avec une **dominante d'expertise** très forte et très profonde (le pied du T) et une **culture générale** sur un grand nombre de sujets connexes (la barre du T). T-shaped va souvent de pair avec la séniorité, car la vision à 360° et la prise de recul s'acquièrent avec l'expérience.

Dans un contexte socio-économique sous tension, il est probable qu'en 2025, on **privilégie ces profils T-shaped seniors**, certes plus onéreux mais qui permettent de constituer des équipes plus frugales, réduisant ainsi les coûts de synchronisation et les besoins en profils annexes.



"It's 30 minutes away. I'll be there in 10!"

Alors en 2025, mettez-vous le focus sur les profils « **T-shaped skills** » comme « **The Wolf¹** » qui seront capables de vous sortir de **n'importe quelle situation ?**

1. Winston Wolfe (Harvey Keitel), un professionnel chargé de résoudre les situations désespérées, permet à deux gangsters de s'en sortir en maquillant une voiture et accessoirement, de sauver un mariage. *Pulp Fiction* ou *Fiction pulpeuse* au Québec (1994) — Quentin Tarantino.

ORGANISATION



iii. Career path Tech

iii. Career path Tech

En 2025, le recrutement et la rétention de **profils tech chevronnés** restent un défi, en particulier pour trois types de rôles :

- les Ops (*hyperscalers* ou clouds souverains) ;
- les profils IA ;
- les développeurs.

Dans notre monde digital où le software est omniprésent, exit la vision traditionnelle qui pousse les profils tech en quête d'évolution vers du pilotage de projet ou du management, et place à l'expertise et aux profils seniors !



Copyright @OCTO Technology

**Enfin un career
path tech
rafraîchissant !**

Les entreprises ont compris le besoin de s'appuyer sur des tech expérimentés. Et dans le même temps, les profils tech souhaitent pouvoir évoluer dans l'entreprise et continuer à développer leur expertise avec leurs pairs.

Il devient donc primordial pour les entreprises pour qui la technologie est un enjeu de faire émerger un **career path** pour les développeurs. Nous pensons que cette initiative va se développer dans les années qui viennent, pour satisfaire des enjeux de **rétention** et **d'attractivité**.

Notre retour d'expérience

Convaincus de l'importance d'un **career path** dans la construction d'une **relation pérenne** entre une **entreprise** et ses **profils tech**, nous vous partageons ici les grandes lignes de celui que nous avons dessiné, inspiré des *Big Tech*, et que nous déployons chez OCTO Technology.

Nous l'avons pensé autour de deux majeures :

- ⦿ l'une axée sur **l'expertise**,
- ⦿ et l'autre sur le **management**.

Nous permettons ainsi à nos *Software Engineers* de se constituer un parcours adapté à leurs ambitions, tout en répondant aux exigences de l'entreprise.



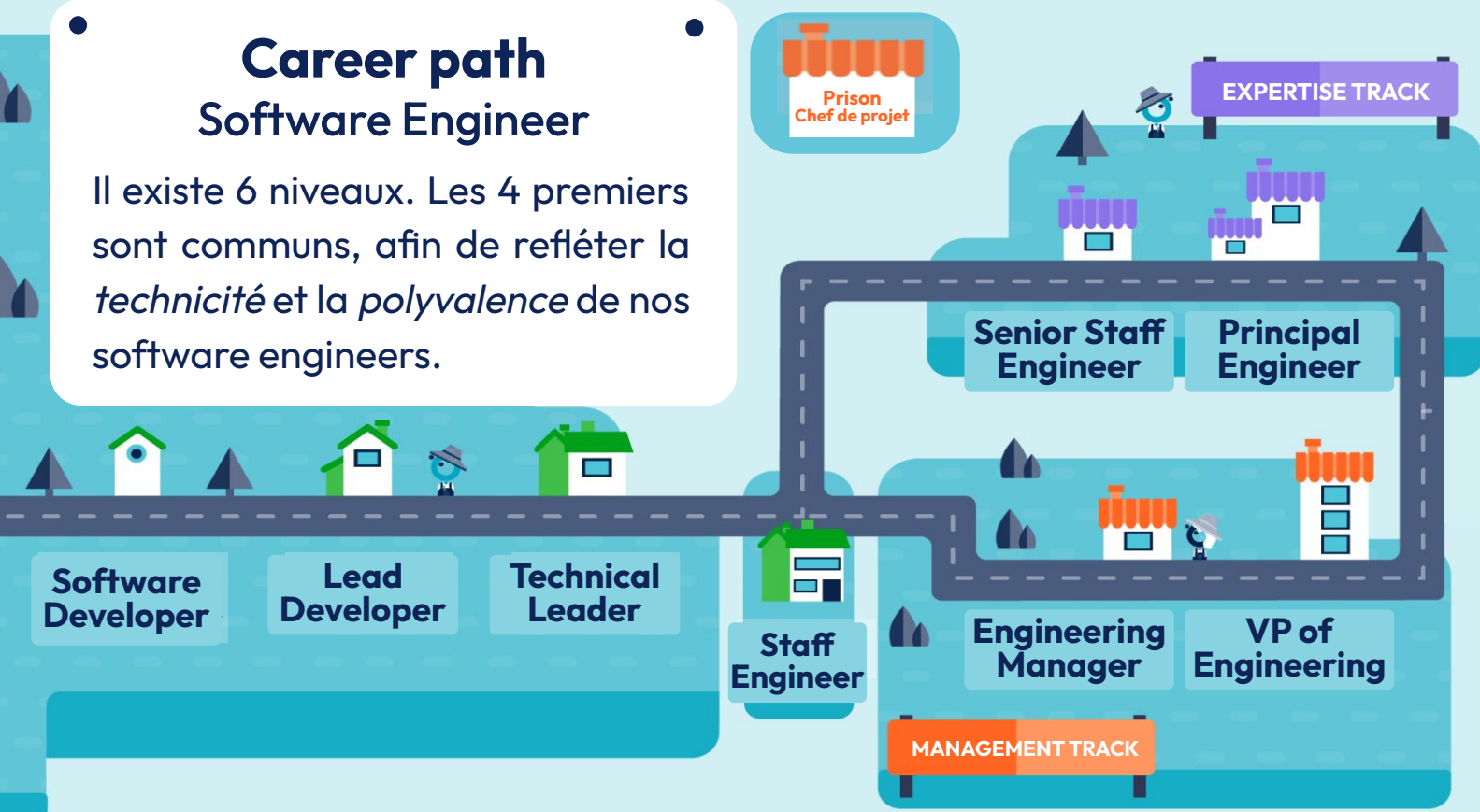
Co-construit par une équipe d'Octos, ce **career path** a vu le jour avec plusieurs objectifs :

- ⦿ Avoir un framework contextualisé qui reflète le **niveau d'exigence** et les **attendus**,
- ⦿ Offrir des perspectives de carrière adaptées à nos **profils tech**, aussi bien **experts** que **managers**, sur des activités de delivery et/ou de conseil.
- ⦿ Recruter et retenir les meilleurs profils tech.

Marie HERNANDEZ,
Software engineering HR Lead

Career path Software Engineer

Il existe 6 niveaux. Les 4 premiers sont communs, afin de refléter la *technicité* et la *polyvalence* de nos software engineers.



Tronc commun

- Software Developer : « *je monte en compétences et je développe mon autonomie* ».
- Lead Developer : « *je maîtrise ma stack technique, je porte des sujets et je commence à transmettre* ».
- Technical Leader¹ : « *j'ai un domaine d'expertise et je suis responsable du bon fonctionnement de mon équipe. Je commence à colorer mon parcours selon mes appétences : expertise, mentorat, management, conseil* ».
- Staff Engineer : « *j'ai travaillé dans des contextes et environnements techniques variés. Je lead des équipes et des sujets techniques conséquents* ».

1. cf. "les 4 piliers du tech lead" dans notre livre blanc [Culture Code](#).

Pour les deux niveaux suivants, nous proposons deux tracks communicantes (on peut facilement passer de l'une à l'autre) à choisir selon les appétences de chacun.

Track Management

- Engineering manager :
« j'organise les équipes pour répondre aux problématiques de nos clients ».
- VP of Engineering : *« je porte des missions de grande ampleur et une communauté conséquente ».*



Track Expertise

- Senior Staff Engineer :
« j'architecture des solutions pour répondre aux problématiques de nos clients ».
- Principal Engineer : *« je fais rayonner OCTO et j'inspire les Octos à travers mes expertises ».*



Au fil des années, quel que soit le profil et les projets professionnels des consultants, nous attendons que leur périmètre de responsabilité et leur impact progressent, en accord avec leurs appétences, les transformations du marché et les besoins de l'entreprise.

Conclusion

En 2025, le **recrutement** et la **rétention** de profils **tech** restent un défi, notamment pour les développeurs chevronnés.

En terme de carrière, certains développeurs ne se projettent pas dans des activités de pilotage de projet, de management ou encore de développement commercial.

Ainsi, dans un monde où *“Software is eating the world”* et où les **profils « T-shaped skills » expérimentés** deviennent **clés**, la question se pose d'offrir un career path dédié.



Alors en 2025, comme **Blondie** pensez vous que le monde se divise en **deux catégories** : un track management et un track expertise, afin d'offrir un **career path** sur mesure aux **profils Tech** ?

1. Blondie (Clint Eastwood) énonce à son ancien complice Tuco (Eli Wallach) “You see, in this world, there is two kinds of people, my friend: those with loaded guns and those who dig. You dig.”. *The Good, the Bad and the Ugly* (1966) — Sergio Leone.



05

SÉCURITÉ.

- i. Zero Trust
- ii. Passwordless



LES CONTRIBUTEURS DE CE CHAPITRE



Antoine
CHANTALOU

Head of Software Engineering
antoine.chantalou@octo.com



Jérémie
KLEIN

Tech T-shaped skills Lead
jeremie.klein@octo.com

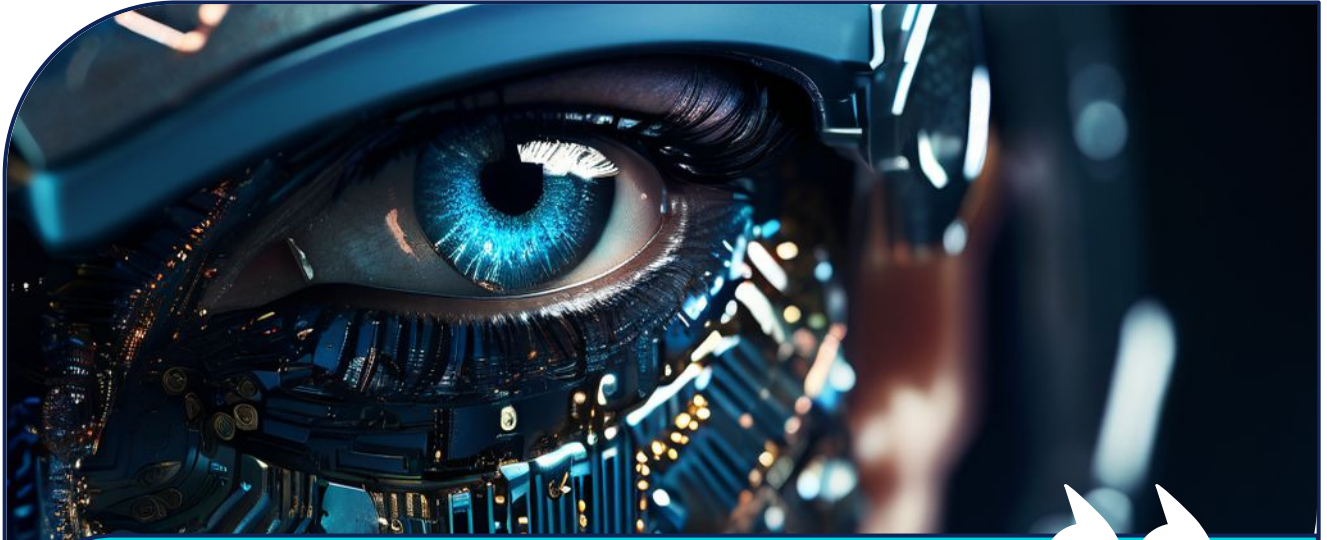


Arnaud
DOUCERAIN

Software security Lead
arnaud.doucerein@octo.com



05. SÉCURITÉ



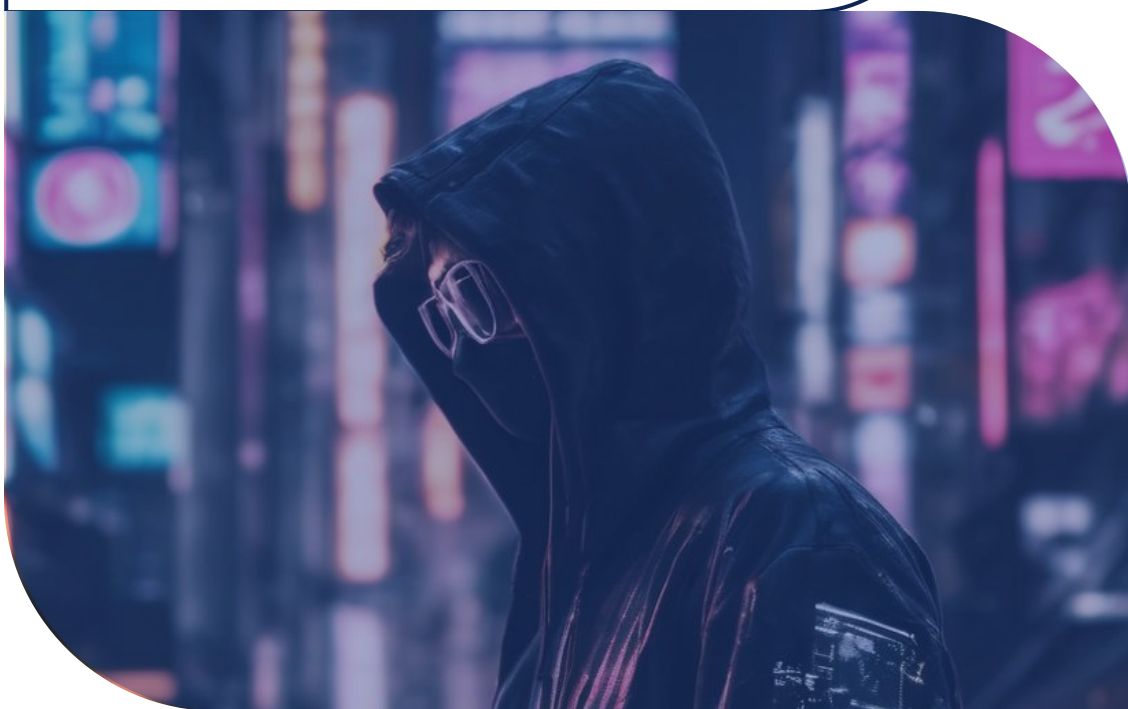
Une sécurisation en profondeur !

Avec les architectures MACH, basées sur des API exposées sur le Web et le cloud-native, combinées à l'archipélisation des lieux de travail post Covid-19, le modèle traditionnel de défense via la **sécurité périmétrique** est mis à mal.

C'est pourquoi deux tech trends de la **cybersécurité** figureront au menu de 2025 :

1. Zero Trust
2. Passwordless

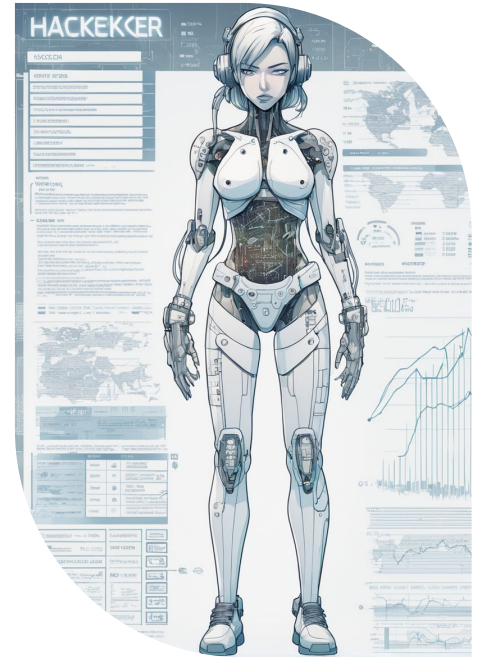
SÉCURITÉ



i. Zero Trust

i. Zero Trust

Si l'hégémonie du **Web** et du **ATAWAD**¹ remettait déjà en question le modèle traditionnel de défense **périmétrique**, avec les architectures MACH basées sur des APIs exposées sur le Web et le cloud-native et l'archipélisation des lieux de travail post Covid-19, les techniques sécuritaires traditionnelles comme les pare-feux, le cloisonnement, la multiplication des VPN ou des solutions logicielles de gestion de la sécurité « **ALL-IN-ONE** » rencontrent leurs limites !



Le modèle **Zéro Trust** est un ensemble de paradigmes :

- ◉ visant à **sécuriser en profondeur l'accès aux ressources du SI** en s'appuyant sur **l'architecture** et des **protocoles d'authentification et d'autorisation standardisés**.
- ◉ considérant que le risque peut venir de **n'importe quel utilisateur du système**, contrairement aux anciens modèles qui considèrent que le risque provient seulement des connexions entrantes.

Nous abordons dans cette section l'impact sur les produits et les architectures que nous bâtissons. En particulier, nous détaillerons l'état de l'art des protocoles d'authentification et d'autorisation : *OpenID Connect*, *OAuth2*.

1. ATAWAD : AnyTime, AnyWhere, Any Device.

2. IAM : Identity Access Management.

Les limites de la sécurité périmétrique

Quelques fondamentaux (quelque peu caricaturés, entendons-nous bien) de l'approche traditionnelle de la sécurité sont :

- Les attaquants n'auront **pas accès à mon réseau/sous-réseau** et, s'ils y arrivent, ils auront des difficultés à basculer des uns vers les autres.
- Une **segmentation réseau** bien faite est donc **efficace** et **essentielle**. Grâce à elle, je peux faire **confiance** à tous les éléments à **l'intérieur**. Je dois en revanche être particulièrement robuste aux **bornes** de mon réseau afin de ne pas faire confiance à un attaquant.
- Je connais à l'avance la typologie de mes utilisateurs. Si je suis une DSI, ce sont mes salariés. Ils **travaillent depuis mes locaux**, sur des **périphériques dont j'ai la maîtrise**.
- Les cas où mes employés devront **télétravailler** sont **rares** et **exceptionnels**. Ces employés en télétravail seront peu nombreux et je trouverai un moyen efficace de leur permettre de travailler sur mon réseau grâce, par exemple, à un **VPN**.



Sécurité périmétrique ?

Quand les douves
et les crocodiles ne
suffisent plus pour
protéger le
château !

Zero Trust, pourquoi maintenant ?

Certaines de ces prémices qui constituent les fondement de la sécurité périmétrique sont mises à mal par trois bouleversements récents :

1. Le travail dans les métiers du tertiaire, et notamment la tech, a basculé vers le travail à distance ou l'hybride depuis la pandémie de COVID-19.
2. La croissance incroyable et la généralisation de l'utilisation du **cloud public**, notamment avec le mouvement des architectures MACH, nécessite d'accepter de sortir une partie de son métier de son réseau d'entreprise. Celui-là même qui était un élément de base de notre stratégie de sécurité périmétrique. On peut bien sûr simuler une appartenance au SI grâce, une fois de plus, à des VPN. Cependant ce sont toujours des solutions de contournement et lorsqu'une partie significative (20%, 30%, 50% ?) de mes nouvelles applications réside dans des clouds publics, est-il vraiment sain de maintenir un régime d'exception ?
3. Les **DSI externalisent** une bonne partie de leur main-d'œuvre à des sociétés de services et/ou de conseil. Il est aujourd'hui excessivement rare pour la DSI d'une grande entreprise de dépasser 50% d'internes. Il est donc compliqué de faire travailler tout le monde avec du matériel sous contrôle. Il est aussi très difficile de définir un profil générique pour les externes qui auront forcément accès à notre réseau.

Ces situations offrent donc un véritable challenge aux postulats proposés. Le lecteur le plus sagace aura peut-être déjà remarqué qu'il semblerait confortable de s'affranchir de certaines limites. Et si l'on arrêta de contrôler nos SI par leurs réseaux ?

Zero Trust Architecture (ZTA)

Zero Trust propose donc une approche différente de la sécurité en reléguant la sécurité par le périmètre réseau au second plan.

L'idée est de renoncer à cette confiance faite par défaut à un acteur, quelle que soit sa provenance (même en interne !). Désormais tout le monde doit prouver qui il est (**authentifié**) et ce qu'il a le droit de faire (**autorisé**) sur une ressource.

L'architecture Zero Trust (ou ZTA pour **Zero Trust Architecture**) est un modèle d'architecture générique, proposé pour répondre à ces nouveaux enjeux.

L'implémentation de référence qui en est à l'origine se nomme **Beyond Corp**. Elle est packagée via Google Cloud Platform, avec des offres comme [Beyond Corp Enterprise](#).



Les 4 piliers de la ZTA

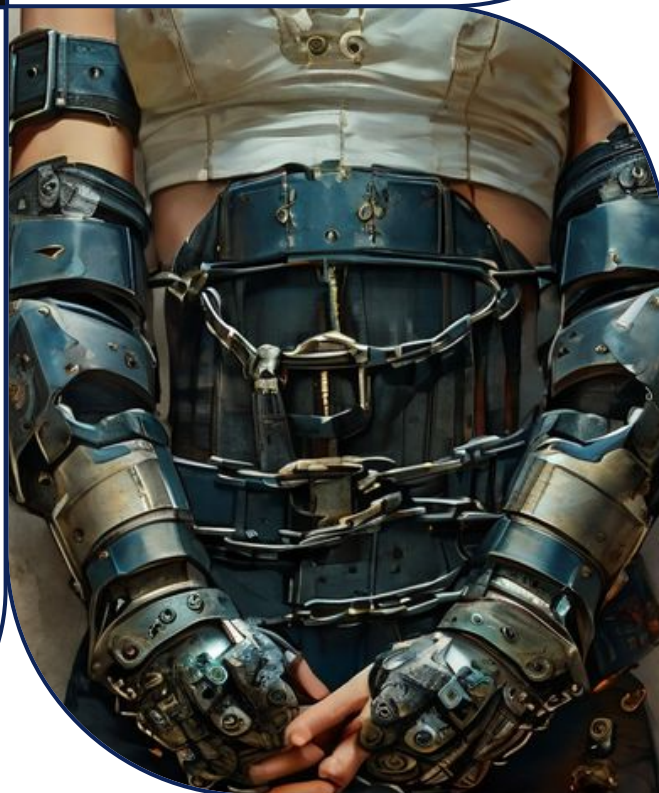


1. Vérifications systématiques et strictes

- **Authentification et autorisation systématiques** (✓ requête)
- **MFA** : vérifier l'identité de l'utilisateur via un *challenge* et en intégrant d'autres facteurs (**biométrie**, question personnelle, etc.)

2. Principe du moindre privilège

- **Accès limité** : octroyer les permissions strictement nécessaires¹ (allow list > deny list)
- **Microsegmentation** : segmentation du réseau en zones plus petites pour limiter les mouvements latéraux et les risques d'intrusion.

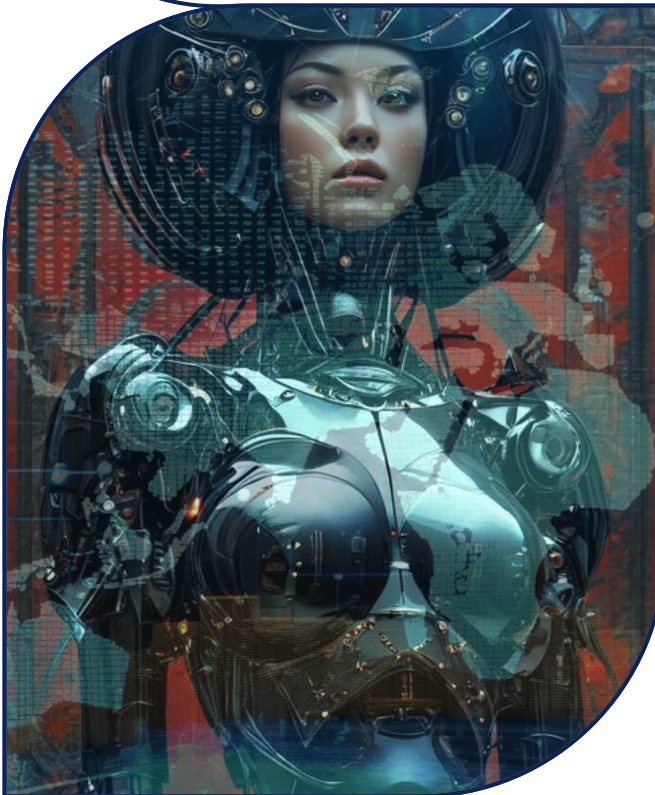


1. nécessaires aux utilisateurs leur permettant de remplir leur rôle. Sachant que ce principe s'applique à une grande variété de paramètres du système : permissions utilisateur, réseau, compte de service, conteneurs, etc. C'est aussi valable pour les accès admins via des solutions de type bastion pour mettre en place une gestion fine des droits et accès avec une piste complète d'audit des actions en production.

Les 4 piliers de la ZTA (suite)

3. Prévention des mouvements latéraux

Prévenir que les **attaquants** se propagent d'un point d'entrée au reste du réseau.



4. Vérification contextualisée

La politique d'accès doit prendre en compte un large nombre d'attributs : identité de l'accédant, **localisation**, analyse comportementale de l'utilisateur, horaires d'accès, etc.

Et si en 2025 on sécurisait (vraiment) nos APIs

Si vous avez lu notre section sur les architectures MACH, vous avez vu qu'un des enjeux réside dans l'exposition d'API. Or, si nombre d'entreprises ont déjà développé des APIs, et continuent d'en développer, l'exposition de ces APIs sur le Web avec une **sécurisation en profondeur** via la **ZTA** n'est pas toujours mise en oeuvre !

En effet, il nous arrive de rencontrer des APIs exposées « *en internes* », pour des partenaires, et les implémentations demeurent basées sur une **sécurité périmétriques**, via des **VPN**, des **filtrage IP**, des **certificats TLS**, etc. Nous rencontrons par ailleurs de nombreux mécanismes de sécurisation basés sur un principe de « *compte de service* », où seuls les contrôles sur l'application appelante sont réalisés et dérogent donc à plusieurs paradigmes de la ZTA :

- **authentification** et **autorisation** de **l'utilisateur** connecté à l'application qui appelle l'API,
- et à fortiori le **MFA** et la **vérification contextualisée**,
- ou encore le principe du **moindre privilège**.

Dans la mesure où une API REST est *stateless*, conformément au principe de la ZTA, ces points de contrôle devraient être réalisés de manière systématique, à chaque appel.

C'est pourquoi en 2025, les standards de l'industrie pour l'authentification et l'autorisation **OAuth2** et **OpenID Connect (OIDC)** seront critiques et devront être systématiquement déployés.

Zoom sur les deux standards clés de sécurisation API : OAuth2 et OIDC

OAuth2

- Ce standard permet à un utilisateur de déléguer une partie de ses droits à une application.
- Il existe plusieurs scénarios pour forger un *access token* quelle que soit la nature de l'application appelante :
 - > **Client Credentials** [Serveur à Serveur]
 - > **Authorization Code** [MPA]
 - > **Authorization Code + PKCE** [MPA / SPA / MobileApp]
 - > **Device Authorization** [Appareil qui n'a pas de clavier (télévision, iot, etc)]

OpenID Connect (OIDC)

- Ce standard autorise une application cliente à récupérer de manière sécurisée l'identité de l'utilisateur. C'est une extension d'OAuth2.
- Le client récupère un access token et un ID token.
- L'ID token est sous la forme JWT¹ et contient des claims qui sont des informations à propos de l'utilisateur et de la façon dont il a été identifié.
- Client et API doivent systématiquement vérifier la signature des JWT.
- La prochaine évolution significative sera les **VCI**² : l'usage d'identifiants numériques qui peuvent représenter des preuves **d'identification physiques** telles qu'un passeport ou un permis de conduire, etc.

1. JWT: JSON Web Token.

2. Verifiable Credential Issuance.

Zero Trust : dois-je y aller et si oui, comment ?

Si **Zero Trust** constitue **l'état de l'art** en matière de **cybersécurité**, la migration de tout un SI vers une architecture Zero Trust tire néanmoins une certaine **complexité**.

Pour déterminer si le jeu en vaut la chandelle, il est pertinent de se poser les trois questions suivantes :

- ⊙ Les utilisateurs de mon SI ont-ils la possibilité de **travailler hors de mon réseau** ?
- ⊙ La plupart de mes applications sont-elles des **applications web et mobiles**, ou la part de celles-ci est-elle en augmentation constante ?
- ⊙ Est-ce que j'utilise le **cloud** de manière significative ou grandissante ?

Si vous répondez « *non* » à au moins une de ces questions, vous n'avez sans doute pas encore un bon rapport *bénéfice/complexité* pour la mise en place d'une architecture Zero Trust. Dans le cas contraire, étudiez sérieusement sa mise en oeuvre.

Vous êtes dans les starting blocks ? Voici quelques axes pour identifier un bon candidat à la mise en oeuvre :

- ⊙ Commencez avec un périmètre restreint, sur des **applications en développement actif**, où l'authentification n'a pas encore été traitée.
- ⊙ Commencez sur des applications **Cloud Natives** et hébergées sur un cloud *public* ou *privé*.

Avec les architectures **MACH**, basées sur des **APIs** exposées sur le Web et le **cloud native**, et avec l'**archipélisation des lieux de travail** post Covid-19, la **sécurité périmétrique** est mise à mal.

L'**Architecture Zero Trust (ZTA)** constitue l'état de l'art de la cybersécurité et sera un enjeu critique en 2025.

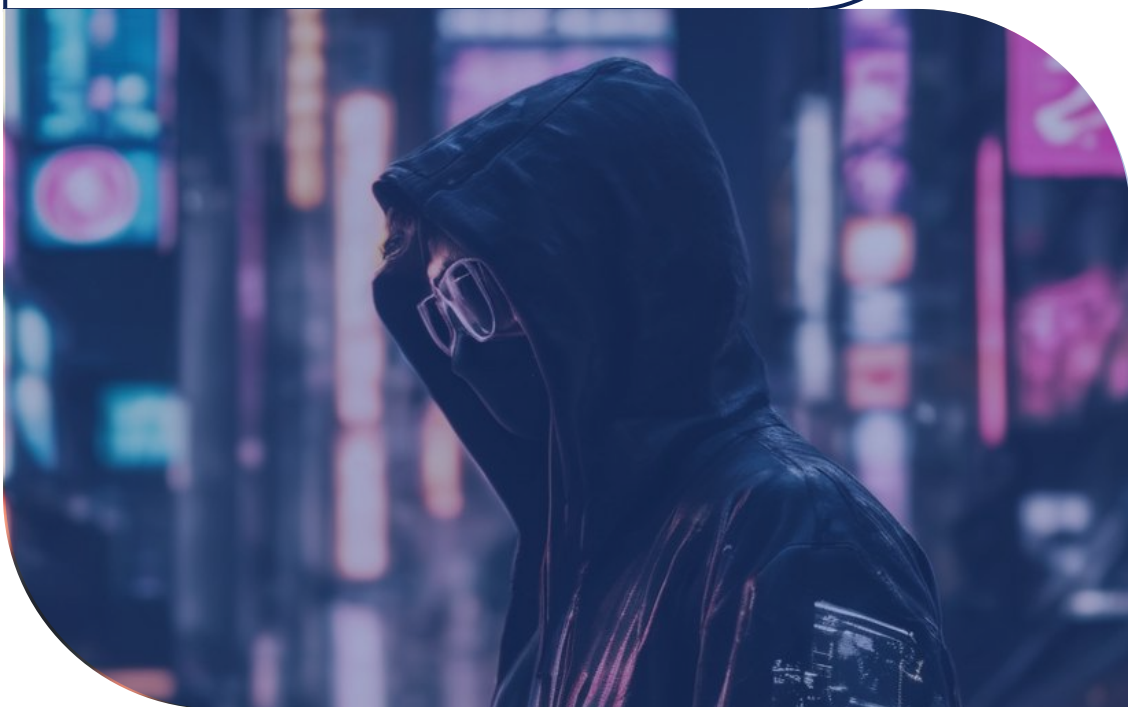
En particulier, pour la sécurisation des APIs, les standards **OAuth2** et **OpenID Connect** ne seront plus une option.



Alors en 2025, embrasserez-vous le **Zero Trust** pour protéger votre système ? Ou prendrez-vous le risque que **Kevin Flynn** puisse s'y introduire avec son programme **CLU**¹ ?

1. Kevin Flynn (Jeff Bridges), un talentueux programmeur de jeux vidéo travaille pour l'éditeur ENCOM. Mais la paternité des jeux qu'il développe est usurpée par un de ses collègues, Ed Dillinger qui, devenu président d'ENCOM, parvient à le faire licencier. À l'aide de son programme CLU : Codified Likeness Utility, il tente de pénétrer le SI ENCOM. *TRON* (1982) — Steven Lisberger.

SÉCURITÉ



ii. Passwordless

ii. Passwordless

De nos jours, la sécurité des **informations personnelles** et **professionnelles** est devenu un enjeu majeur et central de notre société. Rappelez vous par exemple la fuite de données subie par Yahoo.

Le mot de passe, **créature historique** en charge de la protection des données, avec ses défauts inhérents¹, se voit confronté à un nouveau challenger : **passwordless**, qui a pour claire ambition de **révolutionner l'expérience utilisateur** tout en apportant un niveau de **sécurité amélioré**.



Le passwordless, c'est quoi ?

C'est un moyen d'authentification d'un utilisateur sans mot de passe. Au lieu de s'appuyer sur le couple classique *login/mot de passe* pour authentifier l'utilisateur, ce mécanisme s'appuie sur d'autres facteurs tels que :

- ⦿ les dispositifs physiques ;
- ⦿ la biométrie ;
- ⦿ des codes à usage uniques ou par « *liens magiques* », généralement envoyés par email/sms ;
- ⦿ une application mobile pour valider l'authentification.

1. Mot de passe trop simple, utilisation du même mot de passe partout, etc.

Les apports du passwordless

Le passwordless offre de nombreux avantages, autant pour les utilisateurs que pour les services numériques qui décident d'offrir cette possibilité.

Pour les utilisateurs, plus besoin de mémoriser plusieurs mots de passe à rallonge (l'ANSSI recommande des mots de passe de 12 caractères minimum). Si on fait le décompte des différents sites que nous utilisons quotidiennement, en incluant ceux qui ne sont pas reliés en SSO¹, on arrive à beaucoup de mots de passe différents à retenir (en supposant qu'on utilise bien un mot de passe différent pour chacun !). Il existe des password managers permettant la gestion des mots de passe, mais les **dispositifs biométriques** ou les **token** sont en général plus rapides et faciles d'utilisation.

En terme de sécurité, les **gains sont significatifs**. Un mot de passe peut facilement être volé/divulgué à l'aide d'un *keylogger* ou via *phishing*, ou encore deviné (par *brute force* ou *social engineering*). Le fait qu'en 2023, 74% des fuites de données ont été liées à une erreur humaine et ont coûté en moyenne la somme de 4.5 million de dollars, incite fortement à l'emploi de cette technologie.

Un dernier point souvent négligé concerne l'impact sur les équipes support : en évitant d'avoir à gérer la **réinitialisation de mots de passe**, on réduit considérablement la charge de ces équipes qui peuvent alors se consacrer à des activités à plus forte valeur ajoutée.

1. L'authentification unique, souvent désignée par le sigle anglais SSO (de single sign-on) est une méthode permettant à un utilisateur d'accéder à plusieurs applications informatiques (ou sites web sécurisés) en ne procédant qu'à une seule authentification.

Les différentes méthodes d'authentification passwordless ?



La **biométrie** comme une *empreinte* ou la *reconnaissance faciale*, qui est déjà présente sur tous les smartphones récents et certains ordinateurs pour le déverrouillage. Aujourd'hui, elle est utilisée sur smartphone pour authentifier le propriétaire afin de lui donner accès à des applications ou des portions d'application sensibles via 2FA¹.

Les **tokens** ou **clés physiques** (qui utilisent par exemple le protocole FIDO2 qui s'appuie lui-même sur WebAuthn), des dispositifs physiques qui doivent être insérés ou rapprochés de l'équipement afin de permettre l'authentification.



Le code à usage unique (OTP pour One Time Password), généralement envoyé par *SMS* ou *email*, afin de valider l'authentification en utilisant un autre moyen permettant de confirmer son identité.

1. 2FA : two-factor authentication est une méthode d'authentification forte par laquelle un utilisateur peut accéder à une ressource après avoir présenté deux preuves d'identité distinctes.

Enjeux et défis du passwordless

Malgré ses nombreux avantages, l'adoption globale du passwordless n'est pas encore une évidence.

Les entreprises doivent aujourd'hui investir dans ces nouvelles technologies et former leurs employés.

L'achat initial de dispositifs biométriques ou de clés de sécurité peut vite représenter un coût important, en fonction de la taille de la société. Il est donc important, pour un usage interne, de s'assurer que le cas d'utilisation s'y prête.

Pour l'authentification sur des sites publics, mettre en place le passwordless n'engendre pas de coûts matériels, à condition de se reposer sur le smartphone des utilisateurs. Mais il y a cependant un coût de développement et de maintenance de la solution d'authentification passwordless. Notons que la méthode traditionnelle par mot de passe doit néanmoins rester fonctionnelle car elle reste utile en cas de débrayage.

Enfin, sur le plan logistique, la disponibilité de technologies biométriques et de dispositifs de sécurité physique peut varier selon les utilisateurs et les appareils : certains utilisateurs n'ont pas forcément accès à ces méthodes d'authentification alternatives, ce qui peut freiner l'adoption de la technologie passwordless.

Le passwordless est prometteur et sera au menu de 2025. Les tendances émergentes dans le domaine tournent principalement autour :

- Des **avancées biométriques** sur les technologies de reconnaissance vocale et comportementale.
- De **l'intégration avec l'IA** qui va aider à améliorer la précision et la fiabilité des systèmes biométriques.
- De **l'interopérabilité et la standardisation** via des protocoles comme FIDO2 qui vont faciliter l'adoption du passwordless à grande échelle.



Alors en 2025, embrasserez-vous le **passwordless**? Ou bien comme le **M16** dans **Casino royale**, laisserez-vous **007** usurper l'identité de **M¹** parce que vous utilisez des **mots de passe** ?

1. Dans Casino Royale, Bond (Daniel Craig) s'introduit dans l'appartement de sa supérieur hiérarchique M (Judi Dench) et utilise son ordinateur pour identifier le lieu où a été émis le message « *Ellipsis* » sur le téléphone de Mollaka. *Casino Royale* (2006) — Martin Campbell.



| CONCLUSION |

À travers les pages de cet **OCTO PULSE**, nous avons pris du plaisir à partager notre point de vue sur les **Tech Trends 2025** du **Software Engineering**.

Nous nous efforcerons de mettre régulièrement à jour ce document de référence tout au long de l'année et nous publierons une **version Web accessible** très prochainement.

Nous espérons vous avoir aidé à décrypter la **TECH** et préparer au mieux votre **roadmap 2025**, en pointant les sujets de fond, et en allant plus loin que les tendances.

En attendant l'**Épisode #2** qui portera sur... une toute autre thématique ;) Vous souhaitez nous contacter pour approfondir un sujet, discuter ou challenger un point ?

<https://www.octo.com/contact>

*« La tendance c'est le dernier stade avant le
ringard. »*

— Karl Lagerfeld

| L' AUTEUR |

ANTOINE CHANTALOU ●

Excentrique, Rock Star ou encore Grande Gueule, Antoine, avant tout Geek et Architecte, collectionne une panoplie de facettes - et de looks - toujours plus provocatrices au fil des années. Passionné du Web et d'IT depuis 20 ans, il est *partner* et lead de la practice émérite *Software Engineering* chez OCTO Technology.

Depuis 10 ans, il n'a de cesse de vanter l'importance d'une tech bien faite, réalisée par des Doers, avec le mindset OCTO comme gage de la qualité premium attendue par nos clients : « *La crème et pas le lait !* »

RELECTEURS :

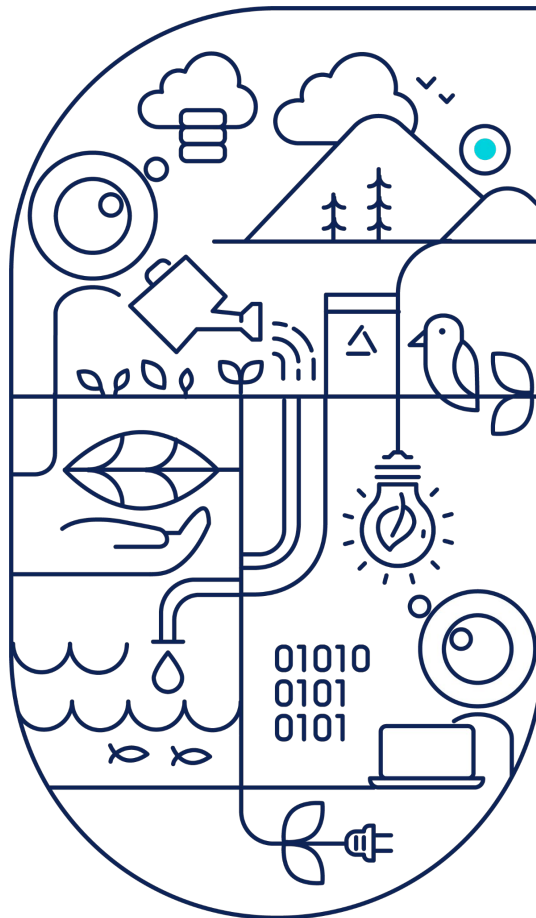
Adrien GRAUX
Antoine MAZURE
Armen OZCELIK
Rémi DORMOY

Relecture formelle :
Alexandra CAUSSARD

Direction artistique :
Camille ROCCHI (IV)



NOTRE WHY OCTO.



DIGITAL THINKERS & DOERS

Dans un **MONDE** complexe aux **RESSOURCES** finies, nous recherchons ensemble de meilleures façons d'**AGIR**. Nous **ŒUVRONS** à concevoir et à réaliser les produits numériques **ESSENTIELS** au **PROGRÈS** de nos clients et à l'émergence d'écosystèmes **VERTUEUX**.



DIGITAL THINKERS & DOERS

**THERE IS
A BETTER
WAY.**