Préambule

Le guide comporte aussi des suggestions non exhaustives d'exercice à réaliser en classe.

Les mises en commun peuvent être réalisée, animée par des outils tels que wooclap, qui permet à tous les apprenants de participer et de contribuer.

# Compétences clef et comportementale d'un·e développeur·se

## Les savoir-faire comportementaux

Ci-dessous, les 6-7 savoir-faire comportementaux sur lesquels travailler en continu lors de la formation (et au-delà). Lors du module OMB, nous avons essentiellement travaillé sur le renforcement de sa curiosité intellectuelle et la pratique d'une veille technologique.

|  |  |  |
| --- | --- | --- |
| **Savoir-faire comportemental** | **Indicateurs** | **Orientations** |
| **Esprit d’analyse** | Compréhension des énoncés  Résolution de problème  Créer des liens entre les informations  Estimation du travail à réaliser  Repérer les difficultés | Développer les capacités d’analyse par exercices et par études de cas, en s’assurant de la bonne compréhension, en anticipant les problèmes, etc.  Se situer dans un processus de développement informatique avec de multiples intervenants |
| **Curiosité intellectuelle** | Auto-formation  Veille technologique  Observation  Apprentissage | Encourager à aller au-delà des modules de la formation par l’exploration personnelle, à pratiquer une veille technologique régulière, à acquérir une discipline dans son apprentissage |
| **Persévérance** | Arriver à complétion des exercices et problèmes rencontrés | Identifier les facteurs potentiels d’échec  Capitaliser sur les expériences d’échec et les transformer |
| **Esprit d’équipe et communication** | Partage des connaissances et de l’information  Cohésion de l’équipe  Langage technique et approprié | Jeu de rôle, persona : prendre la place d’un autre métier lié au développement  Favoriser la communication |
| **Autonomie** | Évaluation de son propre travail (conformité et qualité)  Auto-planification | Initier à la planification et à la priorisation des tâches en fonction d’un cahier des charges donné.  Pousser à dépasser l’écriture d’un code qui fonctionne à un code qui est maintenable par d’autres personnes |
| **Respect des règles / sens de l’éthique** | Respecter des clauses techniques, légales (ex. : enregistrement des données personnelles)  Respecter la déontologie | Initier aux règles déontologiques et au respect des dispositions légales en matière de RGPD |

(issu de : <https://step4-sfc.eu/IMG/pdf/20170512_shopping_list_avec_site_bis_.pdf>)

## Les compétences clefs[C:\Users\ken.dethier\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\269DB054.tmp](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=2ahUKEwiL6rP4jK3lAhUouaQKHWWaCY8QjRx6BAgBEAQ&url=https%3A%2F%2Fkivinen.wordpress.com%2F2018%2F09%2F15%2Fa-short-introduction-to-the-new-key-competences-for-lifelong-learning%2F&psig=AOvVaw1nNdjtpHbX1u5zW2sk8yBl&ust=1571738552872788)

L'Europe propose un canevas pour l'apprentissage tout au long de la vie. Ce canevas, appelé "compétences clefs" (*key competences*), donne des pistes permettant, notamment :

* s'adapter avec souplesse à un monde évoluant rapidement ;
* faciliter l'apprentissage tout au long de la vie.

(source : <https://op.europa.eu/en/publication-detail/-/publication/297a33c8-a1f3-11e9-9d01-01aa75ed71a1/language-en>)

1. **communication dans la langue maternelle :** la faculté d’exprimer et d’interpréter des concepts, pensées, sentiments, faits et opinions à la fois oralement et par écrit ;
2. **communication en langues étrangères :** la faculté décrite au point précédent, à laquelle s’ajoutent des compétences de médiation (résumer, paraphraser, interpréter ou traduire) et de compréhension des autres cultures ;
3. **STEM[[1]](#footnote-2)** compétences en mathématiques ainsi qu’en sciences et en technologies : maîtrise solide du calcul, la compréhension du monde de la nature et la faculté d’appliquer les connaissances et les technologies aux besoins de l’homme (comme la médecine, le transport ou la communication) ;
4. **compétence numérique :** usage critique des technologies de l’information et de la communication au travail, dans les loisirs et dans la communication de tous les jours ;
5. **Apprendre à apprendre :** l’aptitude à gérer efficacement soi-même son apprentissage, à la fois de manière individuelle et en groupe ;
6. **Social et civique :** compétences personnelles, interpersonnelles et interculturelles, couvrant les comportements devant être maîtrisés pour pouvoir participer de manière efficace et constructive à la vie sociale et professionnelle ;
7. **Esprit d'initiative et d'entreprise :** aptitude à passer des idées aux actes. Il suppose de la créativité, de l’innovation et une prise de risques, ainsi que la capacité de gérer des projets en vue de la réalisation d’objectifs ;
8. **Sensibilité et expression culturelle :** expression créatrice d’idées, d’expériences et d’émotions sous diverses formes, dont la musique, les arts du spectacle, la littérature et les arts visuels.

Certaines compétences disposent de leur propre *framework* détaillé, telle que la compétence numérique ([DigComp](https://ec.europa.eu/social/main.jsp?catId=1315&langId=fr&furtherPubs=yes)) et l'esprit d'initiative et d'entreprise ([EntreComp](https://ec.europa.eu/social/main.jsp?catId=1317&langId=fr&furtherPubs=yes)).

Note : on intègre parfois les Arts aux disciplines des STEM pour encourager à la créativité (STEAM).

# Métiers et langages du développement web

## Les métiers du développement web

**[Exercice : demander aux apprenants de citer des métiers]**

**Quelques pistes pour rassembler les réponses des apprenants dans des grandes familles :**

**Client :** personne ou entité qui prend la décision d'acheter un produit ou un service à un fournisseur.

**Équipe commerciale (business et marketing)**

* **Commercial :** le commercial vend des produits ou des services à un client. Il travaille en collaboration avec le marketing et entretient des relations commerciales avec leurs clients.
* **Technico-commercial :** idem, mais augmenté d'une expertise technique du produit ou du service qu'il vend.

**Équipe communication**

* **Chargé de communication (et assimilés) :** il développe la stratégie et les actions de communication vers les publics ciblés, en s'alignant avec la stratégie générale de l'entreprise.
* **Copywriter (concepteur-rédacteur) :** ils écrivent (réécrivent) des textes à vocation publicitaire afin, par exemple, d'améliorer l'aura d'une marque ou de l'entreprise.
* **Designers (graphic, photographie, direction artistique/créative, etc.) :** il étudie et conçoit des produits (maquette, photo, etc.) en harmonisant les critères esthétiques et fonctionnels de celui-ci.

**Équipe de développement**

* **Chef de projet :** il encadre l'équipe travaillant sur le projet informatique en vérifiant la bonne allocation des ressources, le respect des délais, le périmètre, etc.
* **Analyste (business, fonctionnel, etc.) :** il fait le lien entre les besoins "métiers" et leur traduction en application informatique. Il analyse les besoins des utilisateurs et les formalise pour proposer des solutions adaptées.
* **Designers (UI/UX, graphic) :** L'UI/UX designer étudie les interfaces web (homme machine) pour améliorer l'expérience utilisateur.
* **Développeurs :** ils réalisent des applications en se servant des langages de programmation.
  + ***Technical lead* :** développeur ayant une expertise et une expérience significative qui va assurer la cohérence des développements réalisés au sein de l'équipe dans laquelle il travaille.
  + **Développeur *frontend, backend* et mobile :** voir chapitre 7.1
  + ***Database Architect* (DBA) :** l'architecte DB est responsable du design, de la création, du déploiement et de la maintenance des bases de données de l'entreprise.
  + ***Software Architect* :** il prend des décisions sur le design et sur les standards techniques à suivre en matière de plateformes, d'outils ou d'utilisation de langages.
  + **Intégrateur web :** il s'occupe de traduire la maquette graphique en maquette fonctionnelle. Il utilise le HTML et le CSS.
* **Testeurs (QA – *Quality Assurance*) :** il repère les anomalies et les bugs dans une application avant qu'elle ne soit livrée.

**Équipe opérationnelle**

* Développeurs
* Gestionnaire d'exploitation
* *System admin, Network admin, Storage admin, Middleware, Help Desk*, Supervision

## Les langages du développement web

[Exercice : demander aux apprenants de citer des langages utilisés pour le web]

Nous pouvons les classer de plusieurs manières :

* Par type (balisage, programmation, base de données) ;
* Par paradigme (déclaratif, impératif, orienté objet, etc.) ;
* Par niveau (haut et bas) ;
* S'ils nécessitent d'être compilés ou non.

### Bref historique de la genèse des langages web

* 1990-1995 : HTML, CSS et JavaScript sont inventés
* 1996–1999 : La compatibilité entre navigateurs est au début très mauvaise : une même page HTML n'est pas affichée de la même façon suivant le navigateur. Les efforts de normalisation commencent.
* 2000-2004 : Les cadres CSS commencent à émerger. jQuery (JS) naît.
* 2005-2009 : La conformité aux spécifications du W3C est respectée. Le navigateur Chrome prend les devants. Le *responsive design* (interface s'adaptant à l'espace disponible) et les *frameworks* apparaissent.
* 2010–2015 : Les *frameworks* JavaScript naissent, tels que Backbone, Ember, AngularJS, React, Angular, Vue. HTML5 est annoncé.

### Le Javascript : un cas à part

* Langage construit en seulement 10 jours, en 1995. Il est aujourd'hui l'un des langages les plus populaires du web.
* Le JS a la caractéristique d'être interprété très rapidement (par le navigateur).
* Sa popularité grandissante a permis à de nombreux outils de voir le jour, comme le jQuery, Angular, Vue, React, etc.
* Le JS est utilisé aussi bien en frontend qu'en backend (ex. : Node.JS). Cela permet de n'utiliser qu'un seul langage côté client comme serveur.
* Le JS n'est pas utilisé que pour le web : des applications d'ordinateur sont développées en JS, commet Skype.

### Langages de balisage

#### Le HTML

Les langages de balisage permettent de donner une structure sémantique[[2]](#footnote-3) à des documents, comme des textes par exemple. Le langage le plus connu est le HTML, mais il en existe d'autres, comme le LaTeX, utilisé dans le monde académique pour la structuration et la mise en page de documents scientifiques.

Le HTML utilise des balises prédéfinies afin de déterminer dans un fichier texte des éléments tels que les titres, les paragraphes, les citations, les tableaux, les images, etc.

La syntaxe du HTML se compose d'une balise ouvrante "<h1>" et d'une balise fermante "</h1>" (à quelques exceptions près).

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | <h1>Ceci est un titre de niveau 1</h1> | |

**<h1>** : balise ouvrante de l'élément titre de niveau 1.  
**Ceci est un titre de niveau 1** : contenu de l'élément h1  
**</h1>** : balise fermante

Il existe aussi ce que l'on appelle les *Lightweight markup language* (LML, langages de balisage légers) qui utilisent une syntaxe plus simple et facile à lire dans sa forme non interprétée. Le LML le plus répandu est le markdown, utilisé notamment pour la documentation des projets sur Github.

#### Le CSS

Le CSS (Cascading Style Sheets, feuille de style en cascade) n'est pas un langage de balisage à proprement parler : il permet de décrire la présentation des documents HTML, sans pour autant les baliser ou leur donner une structure.

Il permet d'opérer une séparation entre la structure et la présentation. Il est par exemple possible de ne décrire que la structure d'un document en HTML, et de décrire toute la présentation dans une feuille de style CSS séparée. Les styles sont appliqués au dernier moment, dans le navigateur web. Cette séparation fournit un certain nombre de bénéfices, permettant d'améliorer l'accessibilité, de changer plus facilement de présentation, et de réduire la complexité de l'architecture d'un document.

Ex. : <http://www.csszengarden.com/> le même document HTML présenté différemment.

[Exercice : rendre un fichier HTML avec du CSS inline plus lisible en mettant le CSS dans les balises <head> du document, puis dans un fichier externe. Le but de l'exercice est de se rendre compte qu'un langage simple peut devenir difficile à lire sans le formatage adéquat]

### Langages de programmation

Difficile de parler des langages de programmation sans expliquer ce qu'est un paradigme en développement.

Un paradigme est une façon de voir les choses, de se représenter le monde. Appliqué à la programmation, c'est une façon de se représenter la programmation informatique, la vue qu'un développeur a de la façon dont son programme va s'exécuter.

Il existe plusieurs formes de paradigmes en programmation, les plus courants et les plus utilisés sont :

* **Le déclaratif** : on décrit le quoi. Par exemple, dans le HTML, on décrit ce que contient une page. On regroupe dans le déclaratif d'autres types de paradigmes, tel que le fonctionnel.
* **L'mpératif, dont le procédural** : le programme va aller d'une procédure[[3]](#footnote-4) à une autre et cela change son état (ex. : ressemble à une recette de cuisine, qui change l'état des ingrédients au fur et à mesure des "routines")
* **L'Orienté objet** (POO ou OOP) : chaque concept (ou idée) est représenté par un objet. Cet objet possède une structure interne, des caractéristiques et un comportement. L'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, ou de mieux résoudre les problèmes.

En résumé :

* Le procédural est un bon point de départ pour commencer à apprendre la programmation :
  + - Logique plus simple à appréhender ;
    - Résultats plus rapides ;
    - Permet d'envisager les bases, comme les structures de contrôles, qui sont utilisées en OOP aussi ;
* On utilise l'orienté objet :
  + Actuellement le paradigme le plus utilisé en développement web ;
  + Sensé présenter le monde de façon plus naturelle.

### Langages de base de données

Les bases de données les plus courantes dans le développement web sont MySQL et PostgreSQL.

Les bases de données doivent être distinguée des données tabulaires en plusieurs points.

Elles font appel à des technologies différentes, mais ils partagent certaines caractéristiques. Les bases de données impliquent généralement un niveau de traitement technique plus élevé.

#### Principes de base

Les feuilles de calcul et les bases de données sont utilisées pour stocker et gérer des ensembles de données. Le contenu de base d'une feuille de calcul ou d'une base de données est un ensemble de valeurs de données. Là où les tableurs et les bases de données varient, c'est dans la façon dont ils stockent et manipulent les données. Une feuille de calcul stocke les valeurs de données dans des cellules, plusieurs cellules étant représentées dans un système de lignes et de colonnes. Les cellules peuvent faire référence à d'autres cellules, et la feuille de calcul peut inclure des cellules qui effectuent le traitement sur d'autres valeurs de cellules.

Une base de données stocke généralement des valeurs de données dans des tables. Chaque tableau a un nom et une ou plusieurs colonnes et lignes. Une ligne d'une table s'appelle un enregistrement. Un enregistrement unique comprend une valeur pour chaque colonne d'une table. Les bases de données peuvent renforcer les relations entre les enregistrements dans différentes tables.

#### Comparaisons

|  |  |
| --- | --- |
| **Données tabulaires** | **Base de données** |
| **Volume de données** | |
| Les tableurs sont conçus pour analyser les données et trier les éléments de la liste, et non pour le stockage à long terme des données brutes. Une feuille de calcul devrait être utilisée pour " croiser " les chiffres et stocker les éléments d'une seule liste. Les tableurs fournissent les moyens de tenir l'inventaire, de modéliser les données statistiques et de calculer les données. | Pour stocker de grandes quantités de données brutes, il est préférable d'utiliser une base de données. C'est particulièrement vrai dans les cas où deux utilisateurs ou plus partagent l'information. Les bases de données nécessitent peu ou pas de duplication de données entre les tableaux d'information, et les modifications apportées aux données ne corrompent pas la programmation (comme au niveau des cellules d'une feuille de calcul où les calculs sont en cours). En outre, les deux avantages les plus importants de l'utilisation d'une base de données sont la capacité d'accroître l'intégrité des données et la facilité de la communication et du partage des données. |
| **Traitement** | |
| Les tableurs et les bases de données offrent toutes deux une gamme de fonctions de traitement. Vous pouvez ainsi manipuler, trier et filtrer les données. | |
| Pour le traitement des données de base, les tableurs fournissent aux utilisateurs une gamme de fonctions automatisées, accessibles à des personnes qui n'ont pas beaucoup d'expérience technique. | Les bases de données offrent une plus grande complexité en termes de manipulation des données, mais celle-ci doit être exprimée en programmation ou en code SQL. |
| **Accès et présentation** | |
| L'accès à un tableur est parfois limité à une personne à la fois | Avec les bases de données, plusieurs personnes peuvent accéder au même ensemble de données. Ces systèmes gèrent l'édition des données de sorte que deux personnes ne peuvent pas modifier les mêmes valeurs de données en même temps. L'intégrité des données est ainsi préservée, ce qui est plus difficile à gérer lors de l'utilisation de feuilles de calcul. |
| Vous pouvez générer des tableaux et des graphiques à l'aide d'outils logiciels automatisés. | Ces mêmes options de sortie relèvent normalement de l'application qui fournit l'accès aux données. Les bases de données exigent généralement un niveau plus élevé d'expertise technologique. |

### Les niveaux de langage

La notion de niveau désigne la distance du langage par rapport au travail de la machine. Le langage de haut niveau a un plus haut niveau d'abstraction que les langages machines.

De façon stéréotypée, les langages de haut niveau simplifient le travail du programmeur en proposant un langage proche du langage naturel, là où les langages de bas niveau permettent de produire un code plus efficace. La limite d'utilisation des langages haut niveau correspond aux situations où les ressources matérielles sont limitées.

Néanmoins, avec l'augmentation en complexité des architectures des microprocesseurs modernes et l'amélioration des compilateurs, on observe que fréquemment les langages de haut niveau produisent du code aussi efficace que ce qui peut être fait à la main par la plupart des programmeurs. De plus, une abstraction plus haute peut permettre des techniques plus puissantes produisant généralement de meilleurs résultats que leurs équivalents de bas niveau.

Les langages de bas niveau sont utilisés dans : l'informatique embarquée, industrielle, la création de pilotes, de systèmes d'exploitation, voire le développement de jeux vidéo. Dans tous les autres domaines, l'utilisation des langages de bas niveau est contre-productive, parce qu'elle demande au programmeur de consacrer beaucoup plus d'attention, de temps de programmation, et entraîne donc des coûts de production plus élevés, pour réaliser un code équivalent.

### Interprétation et compilation

Dans les langages interprétés, le code source (celui que vous écrivez) est interprété par un logiciel qu'on appelle interpréteur. L'avantage, c'est que le code source n'est pas lié à une machine. Pour autant que l'interpréteur existe, ce code pourra fonctionner sur n'importe quelle machine.

Dans les langages compilés, le code source est d'abord compilé par un logiciel qu'on appelle compilateur, en un code binaire (langage de bas niveau). Le système d'exploitation va pouvoir exécuter en direct les instructions.

# Apprendre à apprendre

## Introduction

Apprendre ne consiste pas simplement à accumuler des connaissances pour les ressortir plus ou moins telles quelles à l'examen. C'est une activité complexe, qui nécessite un accompagnement humain et technique favorable pour se développer en **compétence.**

Ainsi, à la lumière des travaux de la psychologie cognitive et des neurosciences, s'est dégagée **une nouvelle conception de l'apprentissage**, centrée sur l'apprenant, et basée sur les théories de l'activité, **où l'action, la réflexion, et la collaboration avec autrui, sont les conditions essentielles à l'efficacité de l'apprentissage**. Cette conception s'appuie sur notre fonctionnement biologique, cognitif, affectif, et notre nature essentiellement sociale, tout en respectant les multiples différences individuelles.

**Chacun à sa façon d'apprendre**, son propre fonctionnement cognitif. L'important est de découvrir votre propre façon d'apprendre, celle qui vous convient le mieux, et apprendre à la faire évoluer si elle ne mène pas au résultat souhaité.

## Se connaître soi-même

1. **Se percevoir positivement** 
   * S'estimer, avoir confiance en soi pour apprendre dans de bonnes conditions (défier sa critique intérieure : "je ne suis pas assez bon", "ce n'est pas pour moi", etc.)
   * Théorie de l'auto-efficacité (croire en sa capacité à réussir améliore sa capacité à apprendre)
2. **Distinguer motivation et discipline**
   * **Durée :** la discipline est un état d’esprit qui s'installe sur la durée, comme une routine. On peut être motivé à réaliser une tâche aujourd’hui, sans assurance que la motivation se maintienne le jour d'après. L’impact de la motivation n’est que de courte durée.
   * **Stabilité et régularité :** si son esprit est discipliné, entrainé à poser des actions, il sera beaucoup plus facile de progresser dans ses tâches sans avoir à se soucier des facteurs de motivation.
   * **Stimuli interne :** la discipline va amener à prendre des habitudes qui, une fois bien installées, ne coûteront que très peu d'énergie à réaliser ou à démarrer.
3. **Gérer son temps** 
   1. Se fixer des objectifs ;
   2. Définir des priorités ;
   3. Fixer une limite de temps ;
   4. Planifier.

Les écueils courants :

* + Fixer des Objectifs non atteignables, pas réalistes ;
  + Essayer de tout faire en même temps, ce qui peut conduire à une surcharge mentale et cognitive.

1. **Connaitre son style d'apprentissage préférentiel**

Actif <> réflexif

Sensoriel <> intuitif

Visuel <> Verbal

Séquentiel <> Global

1. **Connaître ses intelligences multiples**

* **l'intelligence verbale/linguistique**, généralement favorisée par l'enseignement scolaire, ce qui peut mettre en difficulté ceux chez qui elle est peu développée. Cette intelligence touche tout ce qui attrait au langage, écrit ou oral, et est donc régulièrement employée dans la vie courante. Elle rejoint l’aptitude à penser, avec des mots et à utiliser sa langue, maternelle ou autre, pour partager et exprimer des idées parfois complexes. Cette intelligence est utile pour communiquer, mais aussi pour résoudre des problèmes et créer. Les écrivains en sont bien sûr dotés, mais aussi journalistes, traducteurs, professeurs et avocats pour ne nommer que ceux-ci.
* **l'intelligence logico-mathématique :** les personnes qui ont une intelligence logico-mathématique développée possèdent la capacité de penser de manière abstraite et de calculer, de mesurer, mais aussi de faire preuve de logique. Si vous n’étiez pas bons à l’école pour résoudre des problèmes mathématiques, ça ne veut pas dire que vous ne détenez pas la capacité à faire des liens logiques spontanément. Cette intelligence est somme toute reliée à la capacité d’analyser les causes et les conséquences d’un phénomène et d’expliquer le pourquoi des choses. Les personnes qui en sont dotées, aiment souvent les chiffres, l’analyse et le raisonnement. On retrouve cette intelligence dans une panoplie de métiers, notamment chez les programmeurs, les scientifiques, les ingénieurs et les électriciens.
* **l'intelligence visuelle/spatiale** utile en apprentissage en ce qu'elle peut faciliter la mémorisation et la résolution de problème (voir aussi [les cartes conceptuelles](https://rpn-langues.univ-lille.fr/apprendre/site/co/2-2-3_1.html)). Cette intelligence retrouvée fréquemment chez les artistes est caractérisée par la capacité à mémoriser et visualiser des images. L’intelligence spatiale permet ainsi de se faire une représentation mentale du monde et de se repérer dans l'espace, une capacité qui s’avère fort pratique dans bien des domaines, comme l’architecture, la direction photographique ou même la conception de jeux vidéo.
* **l'intelligence musicale/rythmique**, peut aider à percevoir les sons de la langue étrangère, au travail de l'oral. Ce type d’intelligence est caractérisée par la capacité à apprendre le langage musical, ainsi qu’à penser en rythmes et en mélodies. Elle permet aussi de distinguer et d'apprécier les subtilités de la musique. Évidemment, ce type d’intelligence ou de sensibilité est retrouvé chez les musiciens, mais aussi chez tous ceux qui travaillent de près ou de loin avec la musique et le son.
* **l'intelligence interpersonnelle : l**es personnes qui ont beaucoup d’empathie sont dotées de cette intelligence. Elle permet d’identifier et de comprendre ce que les autres vivent et permet à un individu d’agir et de réagir de manière adaptée. L’intelligence interpersonnelle (ou sociale) fait ainsi appel à la coopération, la tolérance et le discernement. Elle permet également de détecter les intentions d’autrui sans qu’elles soient exprimées clairement. Les personnes qui ont ce type d’intelligence font notamment d’excellents travailleurs sociaux, enseignants, commerçants et médiateurs.
* **l'intelligence intrapersonnelle**, ou la capacité à avoir une bonne connaissance de soi-même : il va sans dire qu'elle concerne de près notre propos sur l'apprendre à apprendre ! En effet, quand cette capacité est insuffisamment développée, on a du mal a prendre le contrôle de sa vie, de son apprentissage, on a du mal a tirer parti de nos expériences et à améliorer ce qui ne va pas, et on cherche des responsables extérieurs à ses échecs !
* **l'intelligence corporelle/kinesthésique : c**e type d’intelligence est liée aux habiletés physique ou à utiliser son corps. On retrouve notamment cette intelligence chez les athlètes, les danseurs, mais aussi les chirurgiens ou des gens qui emploient leur corps pour réaliser ou maximiser une action ou un mouvement. L’agilité et la coordination sont utiles dans bien des domaines et demandent une compréhension, ainsi qu’une appréhension des conséquences de chaque mouvement.
* **l'intelligence naturaliste : c**eux qui sont pourvus d’intelligence naturaliste aiment observer la nature et les choses qui les entourent. Ils détiennent aussi souvent une facilité à apprendre, classer et différencier les choses en catégories. C’est aussi la capacité d’apprécier et de se montrer sensible à la nature et au vivant. Ils se passionnent souvent pour l’environnement et les choses concrètes et observables. Les zoologistes, les botanistes, les jardiniers et les vétérinaires comptent parmi ceux qui présentent ce type d’intelligence.

1. **Connaître ses préférences et sa personnalité**

MBTI (Myers Briggs Type Indicator) est un test de personnalité répandu en Occident. Il est régulièrement utilisé pour diagnostiquer des problèmes de travail en équipe, de communication, ou au niveau managérial.

Le MBTI fonctionne sur 8 préférences disposées sur 4 axes :

* + - D'où tirez-vous votre énergie et où portez-vous votre attention ?
      1. Extraversion : préfère les activités de groupe et se régénère dans les activités sociales ;
      2. Introversion : préfère les activités solitaires et se fatigue plus vite lors d'interactions sociales ;
    - Comment préférez-vous recueillir l'information ?
      1. Sensation (S) : très pragmatique et terre-à-terre. Habitudes et ancrage dans le concret ;
      2. Intuition (N) : imaginatif, ouvert d'esprit et curieux. Préfère la nouveauté à la stabilité.
    - Comment prenez-vous vos décisions ?
      1. Pensée (T) : logique et rationnel. Cache ses émotions et voit l'efficacité comme plus importante que la coopération ;
      2. Sentiment (F) : décide selon ses valeurs personnelles. Sensible et ancré dans le social et la coopération
    - Quel est votre mode de vie idéal ?
      1. Jugement (J) : préfère un mode de vie organisé ;
      2. Perception (P) : préfère un mode de vie souple.

**En résumé**  
Aucun profil, aucune stratégie n'est dans l'absolu meilleure qu'une autre. Les stratégies doivent être reliées à la fois aux caractéristiques cognitives de l'individu, et à la situation d'apprentissage. Les échecs proviennent plus souvent non pas de l'incapacité à utiliser une stratégie, mais de l'incapacité à faire varier les stratégies, selon la nature du problème à résoudre. Une connaissance des styles et des stratégies offre à l'apprenant un plus large répertoire, permettant un choix éclairé et mieux approprié à la situation, et donc une meilleure efficacité.

## Acquérir les bonnes stratégies

* Développer sa curiosité intellectuelle ;
* Fixer son apprentissage et entrainer sa mémoire (ex. : palais de la mémoire) ;
* Améliorer ses capacités à se concentrer (ex. : pomodoro).

## Organiser sa veille technologique

### Les différents types de source

* Sites d'actualité, articles, flux RSS, forums, etc.
* Streaming : youtube, twitch, etc.
* Réseaux sociaux : FB, Twitter, etc.
* Conférences, salons, etc.
* Etc.

### Quelques exemples

**Sites d'actualités et forum en français**

* <https://news.humancoders.com/>
* <https://www.journalduhacker.net/>
* <https://www.developpez.com/>
* …

**Sites web de questions/réponses en anglais**

* Stackoverflow.com
* Quora.com

**Site web d'information généraliste en anglais**

* Medium.com

**Newsletter hebdomadaires en anglais**

* <https://frontendfoc.us/>
* <https://frontendweekly.co/> (plus technique)
* <https://uxdesignweekly.com/>
* <https://mobiledevweekly.com/>
* …

# Gestion de projet

## Définitions :

### Projet

Un projet est un effort temporaire exercé dans le but de créer un résultat, un service ou un produit unique. Il doit avoir un début et une fin. La plupart des projets sont entrepris pour créer un résultat durable et exploitable dans le temps.

### Programme

Un programme désigne un groupe cohérent de projets, dont la gestion coordonnée permet de tirer des avantages qu'une gestion individuelle ne permettrait pas.

### Portfolio

Un portfolio désigne un ensemble de projets, de programmes, de sous-portfolios et d'opérations gérés, en tant que groupe, dans le but d'atteindre des objectifs stratégiques. Il n'est pas lié à un produit défini, ni à une durée.

### Opération récurrente

Une opération récurrente recouvre tous les processus d'activités menés sans limite de temps. Par exemple, un projet livré peut devenir une opération récurrente ; réaliser sa veille technologique est une opération récurrente, etc.

### Activité (ou processus d'activité)

Une activité est un ensemble d'actions ou de tâches permettant d'atteindre un but précis, en transformant des données d'entrée en un produit de sortie. Ces activités peuvent être récurrentes ou non récurrentes.

### SMARTE

Cet acronyme permet de se souvenir de la définition des caractéristiques d'un projet : Spécifique, Mesurable, Atteignable, Réaliste, Temporel, (Écologique/Éthique).

## Les méthodes de gestion de projet

Il existe de très nombreuses méthodes de gestion de projet. Dans le cadre de ce cours, nous allons survoler les deux familles de méthodes les plus répandues dans le développement informatique :

* généraliste (parfois appelée waterfall ou "en cascade") ;
* et l'itératif.

### Pourquoi utiliser ces méthodes ?

Les projets et les activités qui en découlent deviennent de plus en plus complexes, nombreux et interdépendants. Afin d'améliorer le taux de réussite et de complétion de ces projets, une bonne anticipation des ressources (humaines, compétences, temporelles, budgétaires, etc.) est cruciale.

### Gestion de projet généraliste

Ou "modèle en cascade", est une famille de gestion de projet qui met l'accent sur l'anticipation des différentes actions qui devront être menées en séquence (l'une après l'autre, et parfois en parallèle) pour atteindre les objectifs d'un projet.

De manière générale, on classe la gestion de projet en 5 grands groupes de processus :

* **Lancement,** qui s'attache à réaliser des analyses préalables, comme la faisabilité du projet, son alignement vis-à-vis de la stratégie de l'entreprise, identifier les parties prenantes[[4]](#footnote-5), etc.
* **Planification :** définir le périmètre du projet, le planifier dans le temps, le découper en activités successives, lui attribuer des ressources (humaines, etc.), organiser la communication, etc.
* **Exécution,** de réaliser les activités nécessaires à la production du résultat, du produit ou du service.
* **Contrôle et maîtrise,** de vérifier le bon déroulement des activités, en contrôlant l'engagement des parties prenantes, le respect du calendrier, des dépenses, des ressources, de suivre la communication, etc.
* **Livraison,** permettant de clôturer le projet en fournissant un résultat, un produit ou un service utilisable. Il est nécessaire aussi de fournir la documentation et le bilan du projet, afin de servir de base à d'autres projets similaires.

Quelques exemples de méthode reposant sur ce schéma : PMI, Prince2, ISO215000, PM², etc.

### Le cas des méthodes agiles (dont SCRUM)

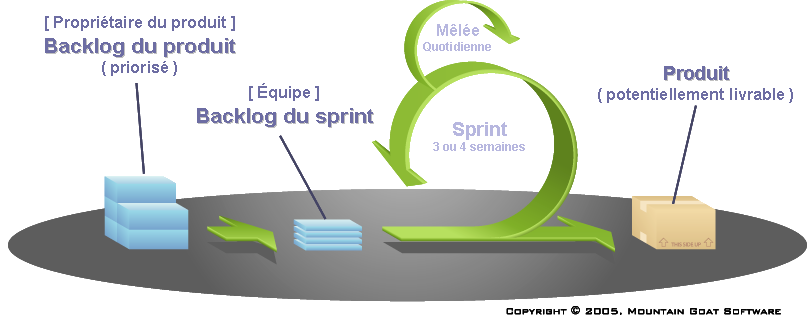
Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles, elles impliquent au maximum le demandeur (client) et permettent une grande réactivité à ses demandes. Elles reposent sur un cycle de développement itératif, incrémental et adaptatif. La progression du projet se fait par versions successives du résultat, produit ou service. L'entièreté des fonctionnalités d'un produit ne doivent plus être identifiées au début, mais pourront être affinées au fur et à mesure de l'avancement des itérations.

Les principes de base sont :

* Comme il est difficile de prévoir toutes les actions à mener pour réaliser un développement informatique, Agile propose un modèle qui va s'adapter de façon rapide aux changements en permettant de livrer, à chaque itération, un produit utilisable ;
* De plus, le client est central dans les méthodes agiles : il participe activement au choix des fonctionnalités du produit qu'il demande.

Le découpage des rôles en SCRUM

* Propriétaire du produit (product owner)
  + Il représente le client et les utilisateurs
  + Il explique et alimente le backlog (liste priorisée des fonctionnalités à développer ou à améliorer), en s'assurant que celui-ci soit visible et compris par l'équipe
  + Il choisit le backlog du sprint
* Maître de mêlée (scrum master)
  + Responsable de la compréhension, de l'adhésion et de la mise en œuvre du travail
  + Assiste tous les membres de l'équipe en maximisant la valeur de ce qu'elle produit
  + En définitive : un rôle de facilitateur dans l'organisation et le suivi des rituels SCRUM, de coaching de l'équipe, de protection de cette dernière (un des rôles clefs d'un chef de projet, ces rôles sont répartis sur toute l'équipe)
* Équipe de développement :
  + Pluridisciplinaire (dev, testeur, architectes, analyste, designer, etc.), sans que les rôles ne soient fixés
  + Son objectif est de livrer un produit utilisable à chaque incrément, potentiellement livrable
  + Collectivement responsable de la qualité du produit, donc pas uniquement des testeurs ou des experts qualité.

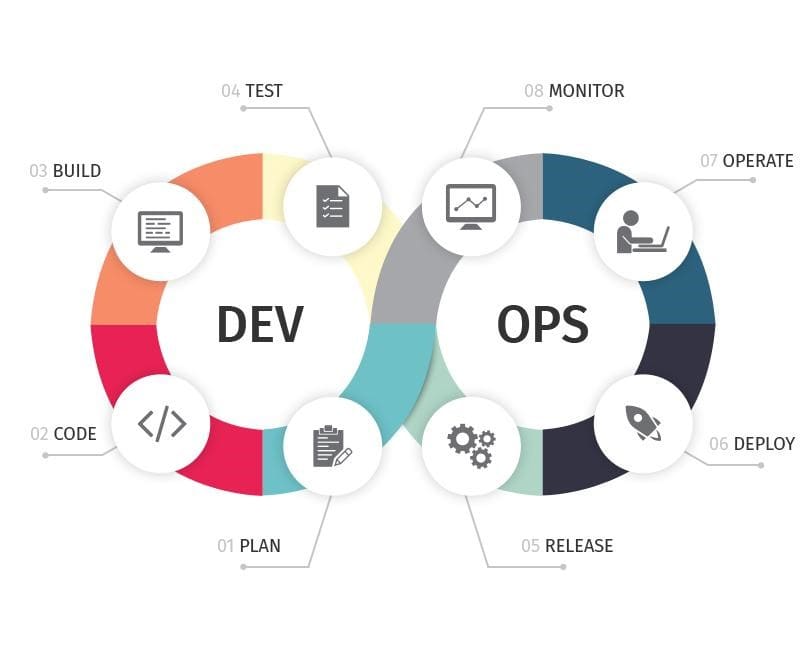


### Autre cas particulier : le DevOps

Le DevOps est pratique technique visant à l'unification du développement logiciel (dev) et de l'administration des infrastructures informatiques (ops).

Apparu autour de 2007 en Belgique avec Patrick Debois, le mouvement DevOps se caractérise par l'intégration de toutes les étapes du développement d'application, depuis le développement, l'intégration, les tests, la livraison jusqu'au déploiement, l'exploitation et la maintenance des infrastructures. Les principes DevOps soutiennent des cycles de développement plus courts, une augmentation de la fréquence des déploiements et des livraisons continues, pour une meilleure atteinte des objectifs économiques de l'entreprise.

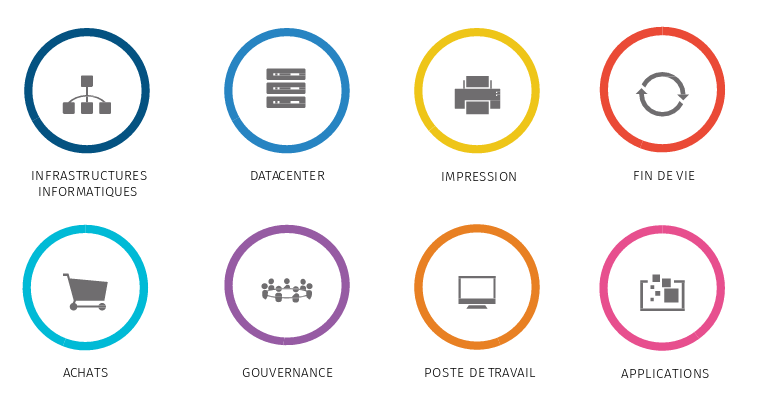
DevOps est la concaténation des trois premières lettres du mot anglais *development* (développement) et de l'abréviation usuelle ops du mot anglais *operations* (exploitation).



## La gouvernance

La gouvernance s'intéresse à la mise en œuvre de bonnes pratiques, permettant à l’organisation que ses investissements informatiques contribuent à :

* accroître la performance des processus informatiques et de leur orientation clients ;
* garantir que les risques liés au système d’information soient sous contrôle ;
* maîtriser les aspects financiers du système d’information ;
* développer des solutions et les compétences en IT, en prévision des besoins futurs.



# Cahier des charges

Un cahier des charges est une première description complète d'un projet : quel est son demandeur, quels sont ses objectifs, comment faut-il les atteindre, etc.

Plusieurs métiers seront amenés à consulter le cahier des charges et il doit présenter, de façon structurée :

* Des informations globales (sur le demandeur, sur le contexte, sur le projet, etc.) ;
* Et des informatiques techniques, détaillées (les descriptions fonctionnelles, notamment).

Il n'existe pas de structure type pour un cahier des charges, ce cahier va dépendre du projet, de l'expérience du rédacteur, de la stratégie adoptée, etc. Un cahier des charges doit cependant pouvoir répondre aux questions des différents métiers amenés à le consulter. Il doit dès lors être rédigé pour favoriser plusieurs niveaux de lecture : un niveau de lecture orienté business et utilité de la démarche (pourquoi demande-t-on cette fonctionnalité ?), un autre orienté technique (comment va-t-on réaliser cette fonctionnalité, avec quelles technologies ?), etc.

L'exercice proposé lors du module OMB a été de se mettre dans la peau d'un rédacteur de cahier des charges en se mettant à la place :

* Des clients et des bénéficiaires de cette nouvelle fonctionnalité ;
* Des métiers techniques amenés à collaborer autour de celle-ci.

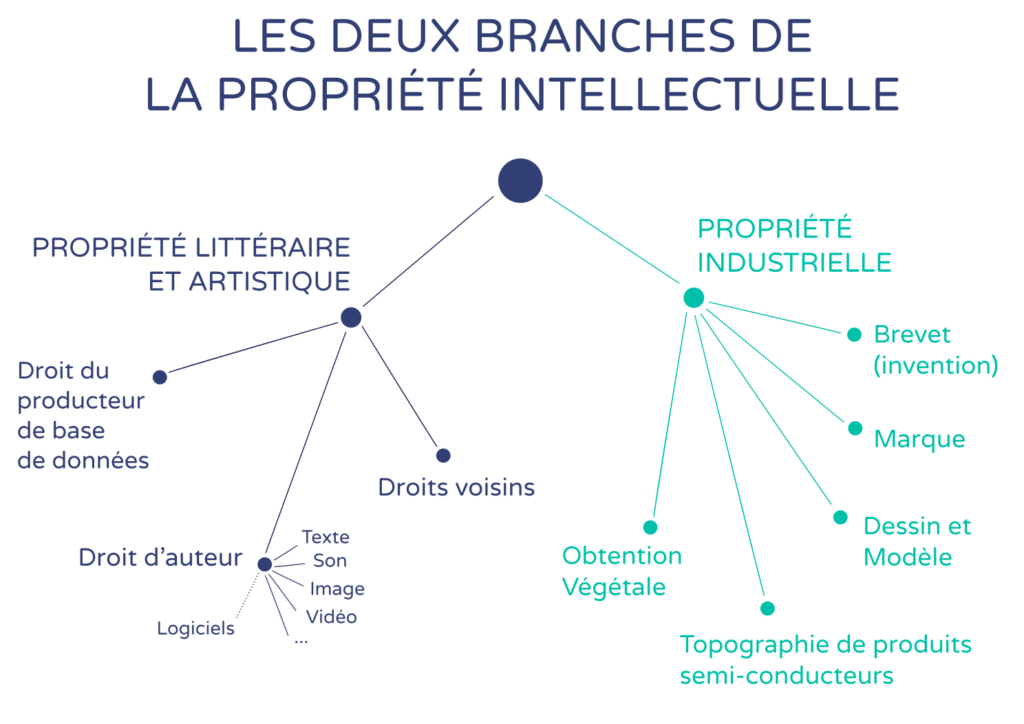
Une bonne pratique est de pouvoir répondre à un certain nombre de questions :

* Qui sont **les cibles et les parties prenantes** de la fonctionnalité demandée dans ce cahier des charges ? Comment sont-elles impactées ?
* Quels sont les **objectifs primaires** de la demande (ex. : améliorer l'existant, ajouter une fonctionnalité simplifiant la navigation dans une interface, etc.) ? Il peut également être utile de décliner les objectifs par cibles.
* Quel est le **message** que vous souhaitez véhiculer ? (ex. : dire, en quelques mots quel est l'objectif principal du point de vue du public le plus important -> cela permet de garder le cap quand on commence à détailler et à descendre dans les fonctionnalités techniques, et de ne pas ajouter des détails qui n'ont pas de sens pour atteindre l'objectif primaire).
* Quelles sont les **contraintes** que vous devez prendre en compte ? (ex. : vous devez utiliser cette technologie server parce que c'est la seule qui est proposée et maîtrisée par votre équipe opérationnelle)
* Quelles sont les **technologies** que vous allez pouvoir utiliser ? Comment communiquent-elles entre elles ? (ex. : comparer des technologies ou des solutions entre elles pour choisir celle qui correspond le mieux à vos attentes)
* Comment allez-vous **évaluer la réussite** du projet ? Quelles sont les mesures que vous allez pouvoir prendre pour vérifier la bonne utilisation de la nouvelle fonctionnalité ? (ex. : vous ajoutez la possibilité de réserver en ligne sur le site web d'un restaurant. Si vous prévoyez à l'avance de mesurer l'utilisation de cette fonctionnalité, vous allez l'implémenter de façon plus complète).
* Comment allez-vous évaluer et **répartir vos ressources** (budget, personnel/compétence, temps) ?

# Déontologie et éthique

## Introduction à la propriété intellectuelle

Remarque : les notes ci-dessous sont un résumé succinct et très simpliste de la propriété intellectuelle du point de vue d'un développeur web.



**La propriété intellectuelle** correspond à l'ensemble des droits qui reposent sur des idées ou des créations, par exemple, un modèle (aspect extérieur d'un produit), une invention, un texte ou une œuvre d'art, etc. On la classe en deux grands groupes : la propriété industrielle et la propriété littéraire et artistique.

**Propriété industrielle :** les deux exemples les plus connus sont les brevets et les marques ("marque déposée"). Quelques exemples de brevets déposés aux États-Unis : achat en ligne en seul clic, l'hyperlien, l'émoticône ";-(", etc.

La **propriété littéraire et artistique** regroupe le droit d'auteur, les droits voisins[[5]](#footnote-6), et le droit des bases de données.

## Le droit d'auteur

Le droit d'auteur est composé des droits patrimoniaux, qui couvrent l'exploitation de l'œuvre, et les droits moraux, qui couvrent la paternité de l'œuvre.

**Droits patrimoniaux :** reproduction, représentation, adaptation. Ils sont cessibles : c'est-à-dire qu'on peut les céder, notamment contre rétribution.

**Droits moraux :** paternité, divulgation, retrait, intégrité. Ils sont incessibles : on ne peut pas les céder.

### Les conditions de protection

Pour pouvoir être protégée, une œuvre devra répondre aux conditions ci-dessous, quelque soit le genre ou le type d'œuvre :

* Critère d'originalité :
  + Ne veut pas dire nouveau ;
  + Doit refléter la personnalité ou l'empreinte personnelle de l'auteur ;

Ex : un listing n'est pas une œuvre originale.

Ex. : une photo, oui.

* Condition de mise en forme :
  + On ne protège pas les idées, mais leur mise en forme. La forme dont l'idée est exprimée peut être soumise à copyright (c'est le brevet protège les inventions et les découvertes).

Le **dépôt d'une œuvre** dans une maison des auteurs confère une date d’antériorité aux œuvres. Celle-ci peut s’avérer déterminante en cas de procès pour plagiat. Il s’agit d’une mesure de précaution (et non de protection) qui propose à l’appréciation de tous les tribunaux un commencement de preuve de l’antériorité de l'œuvre et de l’identité de son auteur.  
-> sacd.be, sabam.be, etc.

### Appartenance d'une œuvre

* En général, elle appartient à la personne physique qui l'a créée ;
* En cas de collaboration, elle appartient à tous les collaborateurs ;
* Dans le cadre d'un contrat de travail dans l'IT : l'employeur possède automatiquement les droits patrimoniaux (exploitation).

### Portée des droits d'auteur

**Droits patrimoniaux**

* Droits exclusifs d'exploitation :
  + Droits de reproduction : exploitation matérielle (ex. : j'achète un tableau ou un logiciel, puis-je le copier ? Non, car je ne suis propriétaire que de son support)
  + Droit de représentation/communication au public : l'auteur est le seul à pouvoir communiquer son œuvre au public, mais il peut autoriser (ou refuser) cette communication à un tiers

Quelques exceptions :

* + - Droit à la citation
    - Communication gratuite dans le cercle familial (ex. : acheter un BR et le regarder en famille)
    - Reproduction dans un but d'illustration dans l'enseignement (exception pédagogique) ou dans une recherche scientifique
    - Caricature, parodie, pastiche
    - Etc.
* Droit à la rémunération

**Droits moraux**

* Droit de divulgation et de retrait : présenter ou retirer l'œuvre au public, l'auteur choisit quand, comment et si elle est prête ;
* Droit de paternité : revendiquer la paternité d'une œuvre, de conserver son anonymat, de la publier sous un pseudonyme ;
* Droit à l'intégrité ou au respect de l'œuvre : l'auteur peut s'opposer à toutes modifications de son œuvre.

De manière générale, les droits sont conservés 70 ans après le décès de l'auteur (ou du dernier collaborateur).

### Protection des programmes d'ordinateur

Le droit d’auteur couvre les programmes d’ordinateur dans ses différents aspects : code-source, code objet (compilation binaire, langage machine), éléments esthétiques, interfaces, matériel préparatoire, structure du programme, etc.

**Le code**

Le droit d’auteur s’applique au code source ainsi qu’au code objet. Il n’est donc pas permis de faire des copies de tout ou partie de logiciels sans y avoir été expressément autorisé.

Il ne suffit donc pas de changer quelques lignes de code pour éviter les problèmes de droit d’auteur.

**L’architecture du logiciel**

L’architecture du logiciel, sa structure, son organisation sont protégées également, ainsi que les routines ou les listes de paramètres.

La simple reprise de l’architecture d’un logiciel pour un autre logiciel est ainsi susceptible de constituer une contrefaçon.

**Le matériel préparatoire et les logiciels en cours de développement**

Le droit d’auteur ne protège pas que le code des logiciels. Il protège également l’ensemble des travaux de conception (dans la mesure où ils sont susceptibles de déboucher ultérieurement sur un programme d’ordinateur). Sont donc protégés les schémas, les organigrammes décrivant tout ou partie de l’architecture du logiciel, les diagrammes de flux, etc.

Par conséquent, copier la structure d’un logiciel dans un autre logiciel est susceptible de constituer une contrefaçon, même si leurs codes sont différents. Cela implique également qu’un logiciel qui est déjà mis en forme, dont l’architecture est suffisamment avancée, est déjà protégé même si le code n’a pas encore été écrit.

**Le look and feel**

Tous les éléments qui relèvent des interfaces-utilisateur, tels que icônes, apparences, écrans des menus, sons, sont également protégés par le droit d’auteur.

### Protection des bases de données

Le seuil d'originalité est impossible à atteindre pour une base de données, car sa valeur vient :

* De son degré d'exhaustivité ;
* De la facilité avec laquelle l'information peut être trouvée par l'utilisateur

Il existe deux types de protection pour une base de données :

* par droit d'auteur : en protégeant la structure de la base de données, si elle est originale ;
* Par Sui generis[[6]](#footnote-7) : en protégeant le contenu (pour protéger, par exemple, l'investissement fait pour réaliser cette base de données contre la copie par des concurrents)

## Les licences d'utilisation

**Contrat d'utilisateur final** (CLUF ou EULA (End User License Agreement) : contrat qui doit être accepté avant d'installer un logiciel ou un service. On y retrouve parfois des clauses à la limite de la légalité.

**Licence publique générale**, plus connue sous l'acronyme **"**GPL" (General public license) : licence de logiciel libre la plus utilisée (pouvant s'appliquer aussi à un livre, un article ou autre création :

* Liberté 0. La liberté d'exécuter le logiciel, pour n'importe quel usage ;
* Liberté 1. La liberté d'étudier le fonctionnement d'un programme et de l'adapter à ses besoins, ce qui passe par l'accès aux codes sources ;
* Liberté 2. La liberté de redistribuer des copies ;
* Liberté 3. L'obligation de faire bénéficier la communauté des versions modifiées.

**Copyleft** : droits des utilisateurs, par opposition aux droits des auteurs.

**Creative commons :** faciliter la diffusion et le partage d'œuvre   
Vidéo : <https://vimeo.com/95488932>

## Accord de non-divulgation

Un accord de non-divulgation (NDA = Non Discolure Agreement) est un accord confidentiel utilisé pour protéger votre propriété intellectuelle. Une NDA contient la définition des renseignements confidentiels, des types de renseignements qui ne peuvent être divulgués et, tout aussi important, des types de renseignements qui ne sont pas assujettis à l'accord. Une autre partie essentielle de la NDA est la limitation de la divulgation et de l'utilisation de l'information et l'indication de la limite de temps de l'obligation des parties.

Deux types de NDA :

* **Entente unilatérale de non-divulgation** - une partie convient de ne pas divulguer les renseignements personnels d'une autre partie.
* **Entente de non-divulgation mutuelle** - les deux parties conviennent de ne pas partager les renseignements de l'autre partie. Le plus souvent, cet accord est utilisé par des entreprises qui partagent des communications protégées.

Un accord de non-divulgation est utilisé :

* Lorsque vous devez partager des renseignements exclusifs avec une tierce partie et que vous ne voulez pas qu'ils deviennent publics.
* Si vous avez accès aux renseignements confidentiels d'une personne et que vous souhaitez définir clairement les modalités de partage de ces renseignements.

En matière de développement logiciel, les informations confidentielles comprennent généralement le code source, les particularités du processus de développement, les détails du projet et des versions, les nouveaux concepts d'applications, une liste de contacts commerciaux, etc.

## RGPD

[](https://www.youtube.com/watch?v=iMD3pWAFXEY)

Vidéo : [Comprendre le RGPD en 5 questions](https://www.youtube.com/watch?v=iMD3pWAFXEY)

Vidéo : [Le RGPD a un an : quel bilan ?](https://www.youtube.com/watch?v=MCg6r9VBXdg)

## Pour aller plus loin :

<https://economie.fgov.be/fr/themes/propriete-intellectuelle>

<https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations_fr?pk_source=google_ads&pk_medium=paid&pk_campaign=gdpr2019>

# Concepts et terminologie

## Définitions et contexte

### Frontend (X75 : interfaces web)

Le développement d’interfaces web couvre la partie visible d’un site web, affichée par l’intermédiaire d’un navigateur, et communément appelé « Frontend ». Le « Backend » couvre, lui, les parties invisibles (accès aux données, base de données, etc.). Le découpage *Frontend* & *Backend* est une abstraction qui permet de simplifier la compréhension d’une application informatique. Dans le modèle client-serveur[[1]](https://reseauifapme-my.sharepoint.com/personal/ken_dethier_ifapme_be/Documents/Projects/Formations/X75-DevInterfacesWeb/X75-DeveloppeurInterfaceWeb_Referentiel-IFAPME-END.docx#_ftn1), la partie client est généralement du ressort du *Frontend*, et la partie serveur, du *Backend*. En réalité, les distinctions ne sont pas toujours aussi nettes.

Le développement d’interfaces web peut aujourd’hui être divisé en deux catégories, liées aux buts poursuivis et aux langages informatiques mis en œuvre :

1. HTML/CSS (et JavaScript interactif basique), avec un intérêt marqué pour le design, les interactions, l’accessibilité, etc., communément rassemblés aujourd’hui sous le vocable « UI/UX Design[[2]](https://reseauifapme-my.sharepoint.com/personal/ken_dethier_ifapme_be/Documents/Projects/Formations/X75-DevInterfacesWeb/X75-DeveloppeurInterfaceWeb_Referentiel-IFAPME-END.docx#_ftn2) » ou d’ergonomie web ;
2. Javascript & PHP, avec un intérêt marqué pour la programmation d’interfaces réactives et la gestion de l’affichage des données issues du *Backend* (par ex. : traitement asynchrone des données, *single page application*, etc.)

L’objet de la formation X75 est de couvrir les deux champs d’application, en privilégiant toutefois la partie programmation. En effet, l’*UI/UX Design* est généralement du ressort des designers et des ergonomes web. Il est cependant important d’avoir une vue complète des activités nécessaires à la construction d’une interface web.

### Applications mobiles (X76)

L’apparition des smartphones a démocratisé l’accès à internet, réduisant ainsi la fracture numérique. Le taux de pénétration de ces outils est nettement plus important que celui des ordinateurs personnels.

Deux *stores* d’applications conçues pour les terminaux mobiles dominent actuellement le marché :

* Android, Playstore de Google :
  + Le plus grand des deux
  + Le moins homogène (smartphones/tablets aux spécifications très différentes (nombreux constructeurs différents), versions différentes d’Android
* iOs, Appstore d’Apple :
  + le plus profitable ;
  + le plus homogène (Apple est l’unique constructeur des iPhone/iPad + obsolescence programmée forcée en software)
  + le plus petit

Le développement d’applications mobiles n’est pas sujet aux mêmes contraintes que les développements d’applications classiques. Le développeur d’applications mobiles doit penser son application différemment en fonction des moyens d’interaction, de la taille de l’écran, de la bande passante, etc. Les librairies de développement mobile existantes tiennent compte de ces contraintes.

Un développeur d’applications mobiles est amené à travailler sur des projets concrets, ludiques et intuitifs qui tirent parti des tendances logicielles actuelles et futures : la mobilité, les réseaux sociaux, la convivialité et l’accessibilité.

### Backend (X77, à partir de 2020)

Le développement d’application backend couvre la partie non visible d’un système d’information, c’est à dire les systèmes qui traitent les données (accès aux bases de données, calculs, intégration, …).

Les applications backend exposent généralement des services via API (*Application Programming Interface*). Ces API permettent à un client d’appeler à distance une application backend via des requêtes et obtenir via la réponse le résultat du traitement backend.

Le découpage *Frontend* & *Backend* est une abstraction qui permet de simplifier la compréhension d’une application informatique. Dans le modèle client-serveur, la partie client est généralement du ressort du *Frontend*, et la partie serveur, du *Backend*. En réalité, les distinctions ne sont pas toujours aussi nettes.

Le développement d’application backend peut aujourd’hui être divisé en plusieurs catégories :

* Applications spécialement conçues pour supporter un site web ;
* Applications spécialement conçues pour offrir des web services réutilisables exposés via des API. Ces applications peuvent être appelées par des applications front end, mais également par d’autres applications backend. On parle alors d’architecture SOA (*Service Oriented Architecture*) ;
* Applications spécialement conçues pour exécuter des traitements lourds, longs et coûteux en CPU. On parle alors d’applications batch qui s’exécutent automatiquement à une certaine heure (par exemple pendant la nuit).

L’objet de la formation X77 est de couvrir les trois champs d’application, en privilégiant toutefois le deuxième type d’application. En effet, ce type d’application est de plus en plus utilisée et demandée en entreprise.

### Fullstack

Le développement *fullstack* touche à toutes les couches de la pile qui constitue une application informatique. De manière générale, le développeur est capable de réaliser les tâches techniques aussi bien du côté frontend que backend, mais également de gérer des projets informatiques avec les méthodes correspondantes (en cascade, itérative, etc.).

## Les commentaires et la documentation

### Introduction

Les commentaires sont souvent délaissés lors du développement, au profit d'autres activités jugées plus importantes, surtout quand il s'agit de respecter des délais.

### Distinguer le commentaire de la documentation

**Documentation (technique) :** descriptif écrit de l'application qui explique comment elle fonctionne, comment elle doit être employée.

Ex. : L’application ne prend pas en charge, dans la version 0.7RC-2, les échanges avec le service X.

**Commentaire :** descriptif écrit dans le code précisant l’implémentation d’une classe, d’une méthode, d’une propriété, ou de toute variable spéciale, quand la nécessité ou le besoin de confort le demande. Un commentaire traduit, en quelque sorte, le code informatique en langage naturel.

Ex. : cette variable est calculée à partir des variables x, y et z. Elle permet de faire xxx.

### Quand doit-on commenter ?

1. En apprentissage :

Au début, il est difficile de savoir quelle partie du code commenter. C'est avec l'habitude que l'on arrive à estimer ce qui doit être commentée ou non. De manière générale, et pour améliorer l'apprentissage, il est conseillé de commenter de façon exhaustive.

1. Pour comprendre ce qu'on a fait :

On se retrouve souvent confronté à un nouveau problème dont il faut trouver la solution. Au bout de quelques recherches, on trouve une solution et on l’intègre. La solution fonctionne et on passe à autre chose.

Trois mois plus tard, on doit revenir dessus car il faut pouvoir y intégrer de nouveaux paramètres ou s’en resservir pour un autre projet, mais on ne se souvient plus de la logique… La petite minute nécessaire à écrire les quelques commentaires salutaires aurait permis d’éviter de chercher à nouveau.

1. Commenter pour expliquer aux autres

On peut travailler en équipe sur un projet et d’autres développeurs risquent d’avoir à intervenir sur votre code pendant que vous faites autre chose. Autre cas de figure : vous développez seul un projet puis un peu plus tard c’est un autre développeur qui le reprend ou qui vous rejoint dessus. Vous pouvez également être en train de développer une application destinée à être partagée avec d’autres.

Le temps passé à commenter évitera aux autres d’être complètement perdus à la lecture de votre code. Ceci est d’autant plus vrai que chaque développeur a son style et sa manière de coder et qui, bien souvent, n’est limpide que pour lui-même…

Tableau commentaires

|  |  |
| --- | --- |
| HTML, une ligne | **<!--** commentaire **-->** |
| HTML, plusieurs | **<!--** commentaire  Sur plusieurs  lignes **-->** |
| CSS, une ligne | **/\*** commentaire **\*/** |
| CSS, plusieurs | **/\*** commentaire  Sur plusieurs  lignes **\*/** |
| VBA | **'** commentaire |
| VBA | Pas de commentaire sur plusieurs lignes |

## Séparation du fond et de la forme

La séparation du fond et de la forme consiste à séparer le message d'un document (le fond) de sa présentation (la forme).

Cette pratique n'est cependant pas une nécessité : elle n'est pas liée à une contrainte technique ou matérielle. Cela dit, vu la complexification des interventions pour créer, par exemple, une page web, le fait de séparer la structure et le contenu de sa mise en forme permet de gagner en lisibilité et facilite donc les éditions futures.

D'autre part, une séparation nette entre les deux permet d'intervenir sur l'un ou sur l'autre sans risquer de créer des erreurs dans l'autre partie.

Ex. : <http://www.csszengarden.com/> -> seul le CSS change entre les pages, la structure et le contenu HTML restent rigoureusement identiques.

## Séparation des préoccupations

On peut appliquer le principe de la séparation du fond et de la forme à une autre échelle : celui des "préoccupations".

***Separation of concerns*** : c'est un principe de conception qui, dans une application, consiste à découper le problème général en plusieurs parties distinctes. L'idée est d'avoir un système bien organisé, où chaque partie répond à un problème précis.

Ex. : Backend/Frontend, cette séparation est abstraite et découpe entre ce qui est visible et non-visible pour l'utilisateur (il faut les deux pour qu'une application marche).

La séparation des préoccupations est organisée autour de "responsabilités". Par exemple, la responsabilité de l'HTML est de structurer et de de donner du sens aux éléments d'une page web. La responsabilité du CSS est d'en décrire le style et la façon dont les éléments sont affichés.

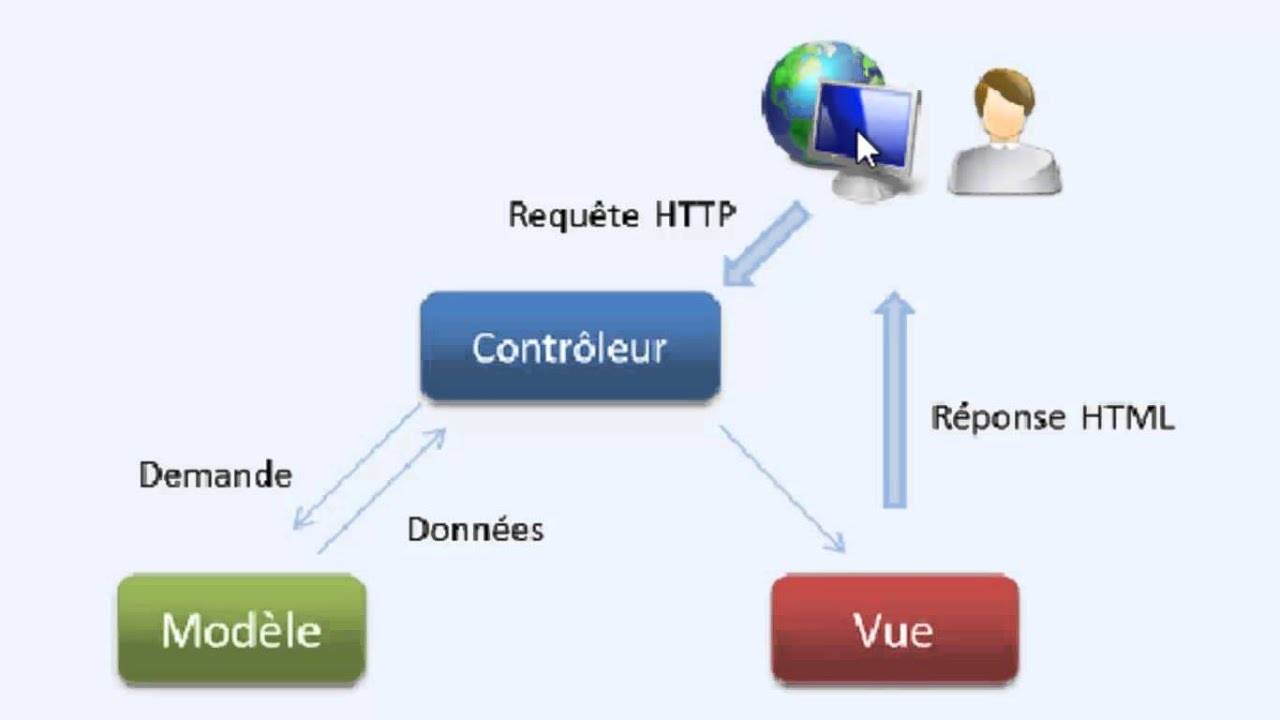
Quels sont les avantages d'une telle séparation ?

1. Peu ou pas de duplication inutile (on ne réécrit pas 4x la même chose) ;
2. Système ordonné, dont les éléments ne s'occupent que d'une seule "préoccupation" ;
3. Facilite la maintenance (on ne risque pas de perturber le fonctionnement d'une préoccupation en modifiant une autre préoccupation).

En pratique, dans le développement web, on va tendre vers un système à 3 *concerns* :

* ce que l'utilisateur va voir ou ce avec quoi il va pouvoir interagir : la **"vue"**
* Ce qui va se charger des évènements (par ex. : le clic d'un utilisateur sur un bouton) et qui va coordonner le flux des données entre les différentes parties de l'application : la **"logique"**
* Ce qui va s'occuper des données, de leur enregistrement et de leur interrogation : les **"données et leur modèle".**

**Le MVC : modèle - vue - contrôleur**



C'est un type d'architecture logicielle très populaire dans le développement d'application web, il est composé du :

* **modèle** : contient les données ;
* **vue** : l'interface graphique avec laquelle l'utilisateur va pouvoir interagir ;
* Et du **contrôleur** : la logique des interactions de l'utilisateur (ex. : "quand l'utilisateur appuie sur le bouton "ajouter un commentaire", l'interface devra afficher une boite supplémentaire, contenant différente fonctionnalité d'édition (ajouter des emoji, formatage du texte, etc.) et un bouton d'envoi.)

Ex. de *frameworks* utilisant ce pattern :

PHP : Symfony, Zend

JS : Angular

Swift

Le MVC poursuit les mêmes objectifs que la séparation des préoccupations et partage les mêmes avantages :

* Plusieurs vues partagent le même modèle ;
* Facilite la création d'une nouvelle interface (puisqu'il "suffit" de faire un nouveau contrôleur et une nouvelle interface, sans toucher au modèle) ;
* Système plus simple à comprendre et à maintenir.

## Dette technique

La dette technique désigne le coût qu'implique de devoir retravailler une solution plus tard, à cause du choix de d'une solution facile, mais limitée, au détriment d'une approche plus solide, mais plus longue (donc plus coûteuse).

Au niveau d'un développeur individuel, ça signifie produire un code qui n'est pas propre, qui n'est pas facile à relire et qui sera difficile à maintenir ou à faire évoluer.

On parle également de "*code smells*", littéralement "code qui sent mauvais" et qui indique qu'il y a un problème. On peut en avoir de plusieurs types :

* Du code dupliqué : du code ayant la même fonction dupliquée à plusieurs endroits -> anticiper l'utilisation des fonctions et des variables ;
* Du code trop compliqué -> "KISS : Keep It Simple, Stupid" ("Garder ça simple et idiot" = ne pas compliquer les choses, pour le plaisir de compliquer)
* Des noms de variable ou de fonctions abscons ou trop court -> nommer les variables correctement pour qu'elles améliorent la lisibilité et la compréhension du code
* Lignes de code trop longues et illisibles -> écrire son code de façon structurée, en suivant les conventions usuelles.
* Etc.

## REGEX/Grep

Une expression régulière (REGEX : **reg**ular **ex**pression) permet de chercher des motifs dans des chaines de caractères. Il est très utilisé en informatique notamment pour :

* Réaliser des opérations de "rechercher/remplacer" automatisés (remplacer tous les tirets par des virgules, extraire juste l'année dans une liste de dates, etc.) ;
* Vérifier qu'un utilisateur a rempli le champ d'un formulaire avec les données attendues (ex. : une adresse email)

En résumé, vous allez l'utiliser pour extraire de l'information depuis des fichiers texte, comme du code informatique, des logs, des données tabulaires, etc.

<https://regexone.com/> exercices.

## Les biais cognitifs

Notre cerveau a mis en place des stratégies pour résoudre les problèmes qui lui sont présentés, notamment par le recours à des raccourcis spontanés et inconscients. Ces raccourcis peuvent parfois nous faire mal interpréter les choses, c'est pour cela qu'on parle de "biais" cognitifs. Ils peuvent également se transformer en leviers pour influer sur nos processus décisionnels. Ils sont très utilisés en marketing et en construction d'interfaces homme-machine.

### Quelques exemples de biais cognitifs

#### Effet de halo

L’effet de halo affecte la perception des gens ou d'un groupe. Une caractéristique jugée positive à propos d'une personne ou d'une collectivité a tendance à rendre plus positives les autres caractéristiques de cette personne, même sans les connaître (et inversement pour une caractéristique négative).

Ex. : il est "beau", donc il doit être "intelligent".

#### Les coûts irrécupérables

En économie comportementale et dans l'analyse de la décision, les coûts irrécupérables (*sunk cost* en anglais) sont les coûts qui ont déjà été payés définitivement ; ils ne sont ni remboursables, ni récupérables par un autre moyen. De façon uniquement rationnelle, ces coûts ne devraient pas influer sur les choix qui sont réalisés après qu'ils aient été engagés.

Ex. : un spectateur dans une salle de cinéma qui trouve le film très mauvais et s'ennuie hésitera souvent à quitter la salle avant la fin du film, pour ne pas gâcher l'argent qu'il a dépensé pour son billet. Mais si un ami lui a donné un billet gratuitement parce qu’il ne pouvait pas se rendre à la séance, le même spectateur n'hésitera alors généralement pas à partir.

De façon strictement rationnelle, les deux décisions sont pourtant exactement équivalentes, puisque le coût financier de quitter la salle est nul dans les deux cas, et l'intérêt du temps qu'il va passer à l'extérieur par rapport à celui passé à regarder le film jusqu'à la fin est lui aussi le même.

#### Cas particulier : culte du cargo

Le "culte du cargo" est un ensemble de rites qui apparaissent dans la première moitié du 20e siècle chez les aborigènes, en réaction à la colonisation de la Mélanésie. Il consiste à imiter les opérateurs radios américains et japonais commandant du ravitaillement (distribués par avion-cargo) en espérant déboucher sur les mêmes effets, sans en comprendre le fonctionnement.

Ex. : en informatique, on le retrouve souvent dans le développement : dans le choix d'une nouvelle méthode de développement, d'une technologie, sans en comprendre l'utilité ni les faiblesses.

1. Pour Sciences, Technologies, Engineering and Mathematics [↑](#footnote-ref-2)
2. Donner du sens. Dans le cas d'un langage de balisage : déterminer qu'une portion de texte est un titre, lui conférant un sens différent que s'il avait été déterminé comme paragraphe. [↑](#footnote-ref-3)
3. séquence d'instructions ou de routines [↑](#footnote-ref-4)
4. Une partie prenante, en gestion de projet, est une personne, un groupe ou un organisme qui peuvent affecter, être affectés ou se sentir affecté par un quelconque aspect du projet. Quand vous travaillez sur un développement web, les parties prenantes sont vastes : le client, les publics visés, la hiérarchie, les partenaires, les métiers connexes qui vont collaborer au développement (voir chap. sur les métiers du développement), etc. [↑](#footnote-ref-5)
5. Servent à protéger la contribution artistique ou financière investie dans une création littéraire ou artistique, tel que les artistes-interprètes, les producteurs (musique, film, etc.), etc. [↑](#footnote-ref-6)
6. qu'on ne peut pas confondre avec une autre [↑](#footnote-ref-7)