

Origines et Évolution de l'Informatique

De la gestion de l'information à son automatisation

Dorian Blanchard

Mémoire de fin d'études pour la validation du diplôme résultant de mon parcours en
Mastère Architecture des Systèmes d'Information à l'École Hexagone.

Bienvenue

Ce document PDF a été généré avec du code Markdown édité en temps réel à l'aide du logiciel Typora. Les textes violet foncé soulignés sont des liens de navigation. Les notes de bas de page et références bibliographiques seront indiquées avec des annotations cliquables comme suit. ¹

L'usage du pronom impersonnel « on » est proscrit, et je souhaite limiter l'usage d'annexes et de lexique / glossaire pour expliciter mon texte. Si vous êtes un(e) relecteur(ice), et que vous constatez quelque chose de tel ou ne comprenez pas une certaine partie, je vous remercierai de me l'indiquer afin que je puisse essayer de retravailler le texte pour le rendre explicite et ainsi éviter aux futurs lecteurs de devoir aller chercher une définition, ou une page tierce afin de poursuivre sa lecture en ayant compris. Si un terme de vocabulaire vous pose problème ou vous a demandé l'usage d'un dictionnaire, j'adapterai le texte pour l'explicitier si possible. Autrement, je le rajouterai au lexique, ou l'expliquerai en note de bas de page. Ne négligez pas la préface, l'introduction ni la problématique, sinon vous risquez de ne pas comprendre le développement.

Sommaire

Sous-titre	Page
0.1 — Bienvenue	1
0.2 — Sommaire	2
0.3 — Résumé	4
0.4 — Préface	4
0.5 — Remerciements	7
0.6 — Introduction	8
0.7 — Comprendre la problématique	9
1.0 — Histoire	11
1.1 — Préhistoire	11
1.2 — Antiquité	13
1.2.1 — Mésopotamie	13
1.2.2 — Grèce	15
1.2.3 — Égypte	17
1.3 — Moyen-Âge	19
1.4 — Époque moderne	24
1.4.1 — Révolutions informationnelles	25
1.5 — Époque contemporaine	29
1.5.1 — De la mécanique à l'électronique	29
1.5.2 — Genèse de la programmation informatique	34
1.6 — Guerres mondiales	37
1.6.1 — Guerre informationnelle et anticipation	37
1.7 — Guerre froide	43
1.7.1 — Genèse de l'informatique moderne	43
1.7.2 — Temps réel	46
1.7.3 — Âge d'or d'IBM	48
1.7.4 — Émergence de l'Intelligence Artificielle	50
1.7.5 — Du circuit électronique au microprocesseur	52

Sous-titre	Page
1.7.6 — Course à l'espace	53
1.8 — Bilan passé	53
2.0 — Pratiques actuelles	54
2.1 — Genèse d'internet	54
2.2 — Interfaces graphiques et périphériques de pointage	56
2.3 — Ordinateurs personnels et jeux	57
2.4 — Logiciel libre, open source, et cadriciels	58
2.5 — Chiffrements actuels	59
2.6 — Décisions et gestion de projet	60
2.7 — Apprentissage et développement personnel	64
2.8 — Metagame	69
2.9 — Bilan actuel	70
2.10 — Concepts et outils de programmation	71
2.10.1 — Réalisation et mise en ligne d'un site web	72
2.10.2 — Langage logique	76
2.10.3 — Principes SOLID	80
2.10.4 — Problèmes et solutions	80
2.10.5 — UML	82
3.0 — Comment repenser la gestion de l'information pour moderniser l'expérience développeur ?	83
3.1 — A priori	86
3.2 — A posteriori	89
3.3 — Conclusion	91
4.0 — Postface : Rétrospective et métacognition	93
5.0 — Annexes	97
5.1 — Bibliographie	97
5.2 — Webographie	98
5.3 — Code	100
5.4 — Lexique	101
5.5 — Table des illustrations	102

Résumé

Dès la préhistoire, l'humanité avait déjà des méthodes de gestion de l'information. L'histoire de l'information est celle de l'ensemble des individus de notre espèce, qui y contribuent durant leurs vies au travers de leurs actes, lectures, écrits, et dires. Cet ouvrage, conformément à son titre et aux recherches qu'il a nécessitées, énonce donc le passé, le présent, et mon anticipation concernant le futur de la gestion de l'information. L'étude des connaissances existantes sur ce sujet vous replongera sûrement dans vos cours d'Histoire. Cet état de l'art mettra en situation le présent avec un abrégé allant de la préhistoire jusqu'à l'état actuel de l'informatique en tant que science de la gestion informationnelle. D'après mes 6 ans d'expérience professionnelle en tant que développeur, j'ai pu voir en quoi cette discipline récente permet désormais de traiter et automatiser une myriade de tâches pénibles, mais aussi que sa pratique en elle-même reste fastidieuse. C'est pourquoi j'en ai fait le cœur de ma problématique, et que la dernière partie se nomme « Comment repenser la gestion de l'information pour moderniser l'expérience développeur ? ». À l'aide des parties précédentes analysant son origine, la conclusion proposera ainsi une solution informatique innovante, combinant des technologies existantes pour se simplifier elle-même en utilisant de la rétroaction, et de la métaprogrammation à travers une interface graphique simple et agréable.

Préface

Avant que je ne me présente et en vienne à parler de *front* et *back* pour mes intitulés de postes, il s'agit des deux principaux métiers de la réalisation de logiciels. Ils se définissent respectivement et simplement comme le service client et l'administration, comme la scène et les coulisses, comme la gestion de l'affichage utilisant le matériel du client d'une part, et ce qui se passe dans les boîtes noires logiques que sont les serveurs de l'autre.

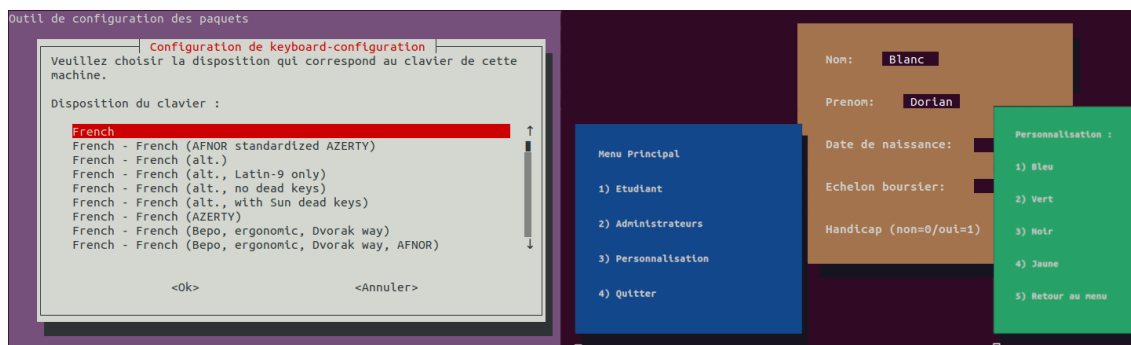
Un développeur *full-stack* est quant à lui, responsable des deux.

Pour pouvoir contextualiser cette œuvre qui se questionne sur l'évolution de l'informatique, je vais brièvement vous parler de la mienne en abordant surtout mon apprentissage de cette discipline. Vous pourrez ainsi mieux comprendre mon identité et les raisons qui m'ont poussé à effectuer des recherches sur l'Histoire de l'informatique ou à trouver des solutions pour améliorer mon expérience développeur.

Après l'obtention de mon baccalauréat scientifique en 2015, débarquant alors en DUT informatique, je n'avais programmé qu'en BASIC sur une calculatrice Texas Instrument. À cette époque, je n'étais qu'un adolescent de 16 ans ayant sauté la grande section et le cours primaire pour avoir su lire à 2 ans et demi grâce au logiciel éducatif Lapin malin. Je débarque donc à Clermont-Ferrand pour mes études supérieures, découvrant l'indépendance dans un 9m²

fibré du [CROUS](#) pendant 2 ans. Je jouais beaucoup aux jeux vidéos et ne fournissait que peu d'efforts dans ma scolarité. Malgré cela j'obtenais toujours la moyenne, bien que cette dernière chutait année après année. Je n'étais évidemment pas prêt à la complexité et la rigueur du C ou de Bash. Jusque là, mon apprentissage de cette discipline sur ma calculatrice était purement empirique. Lors de la première année de [DUT](#), l'enseignement de la programmation nous a été introduit sur papier, en commençant donc par la théorie que les étudiants n'avaient pour la majorité jamais pratiquée et donc du mal à conceptualiser. Comme je n'avais pas encore l'habitude de travailler assidûment, j'ai logiquement été noyé par la quantité d'information et ne les comprenais souvent qu'après les évaluations, lors de la correction, me rendant compte de mes erreurs et réajustant ma version de ce qui était correct en fonction. J'étais tellement largué en contrôle que lorsque je me suis approprié la théorie en réalisant 80% d'un projet de groupe en langage C, j'ai pourtant eu une note moindre que mes camarades, car mon niveau au moment d'un précédent partiel théorique était relativement faible.

[Ce projet est toujours disponible sur mon GitHub](#), il s'agit d'un gestionnaire de résidence étudiante. Moi qui haïssais alors la ligne de commande repoussante et très peu conviviale ou facile à utiliser, étais alors le seul étudiant de la promotion à avoir eu la fantaisie de réaliser les affichages en changeant la couleur de fond et du texte. J'ai ainsi simulé une interface graphique plus agréable, à l'instar des premiers BIOS et des interfaces console interactives que nous avons toujours aujourd'hui comme `dpkg-reconfigure` sur Linux, qui est soit dit en passant, car j'ai eu le cas récemment : le meilleur moyen de changer la langue de son clavier pour passer de QWERTY à AZERTY de façon permanente en utilisant la ligne de commande.



F1 : Comparaison dpkg keyboard-configuration VS residence-manager

J'ai rapidement découvert les compilateurs, qui indiquent les erreurs tant bien que mal, qui ont parfois eux-mêmes des erreurs ou manques de précisions, nécessitant de comprendre pleinement son fonctionnement ainsi que son propre code pour arriver à trouver et corriger le problème. Cela peut être décourageant pour beaucoup, et c'est d'ailleurs certainement une des raisons pour lesquelles une bonne partie des élèves abandonnent en première année.

En deuxième année, j'ai dirigé et développé en grande partie un projet libre de fin d'année. Comme beaucoup d'étudiants en informatique, je voulais réaliser un jeu, et me suis lancé dans son développement en utilisant C# et MonoGame, un outil logiciel facilitant la création d'un jeu en fournissant de quoi afficher des rectangles, remplis de couleurs ou d'images, et les règles

à leur appliquer à chaque itération, avant mise à jour de l’affichage. Pour compléter les fonctionnalités fournies, j’ai développé mes propres outils logiciels pour gérer la vitesse et les collisions entre les entités. À cet effet, le calcul d’angles et de distances, la trigonométrie ou le théorème de Pythagore m’ont tous été très utiles.

Après l’obtention de mon DUT, je n’ai pas trouvé d’alternance et donc pas été pris en licence professionnelle. Je me suis donc dirigé vers une deuxième année de licence en informatique dans le but de réaliser un master. Lors de cette année, il nous a été demandé de réaliser un projet de client-serveur employant des sockets pour une communication en temps réel. J’ai donc pu réutiliser mon projet de C et les connaissances de mon DUT avec bien entendu la technique pour simuler une interface graphique dans un environnement console, créant avec succès [Ohen, un jeu multijoueur en ligne](#). J’ai de même pu réutiliser mes compétences en développement de jeux vidéos grâce à C# et MonoGame pour la réalisation du fameux [Jeu de la vie de John Horton Conway](#). Ce projet est aussi disponible sur mon GitHub dans une [compilation de mes projets MonoGame](#), avec un [installeur Windows fonctionnel](#).

Pour autant, je me suis vite rendu compte qu’il me manquait des bases pour aller au bout de ce cursus, notamment le vocabulaire mathématique enseigné dans la première année de licence que je n’avais pas réalisé, et la motivation nécessaire pour réussir dans cette formation plus théorique, nécessitant par la suite d’entrer en master pour aboutir un Bac + 5. J’ai donc décidé de ne pas poursuivre dans cette voie et de me rabattre sur une licence professionnelle web afin d’avoir un diplôme professionnalisant. Après un an d’incertitudes et de remises en question, j’ai pu signer un contrat d’alternance en tant que Développeur Full-Stack chez l’éditeur de logiciel [CRM](#) nommé SoEMan Group. Cette expérience professionnelle m’a permis de mieux comprendre [SQL](#) et [PHP](#) que pendant mes cours. J’ai même pu reprendre un projet de refonte graphique de l’application web pour la rendre fonctionnelle sur toutes tailles d’écrans. Ce projet a ensuite lui-même été repris et aboutit. J’ai apprécié cette mission de développement front-end, car j’ai moi-même maqueté les interfaces avant de les réaliser, pouvant voir concrètement l’effet du code sur l’affichage, sans avoir cette sensation de boîte noire incompréhensible que procure le développement de la gestion de l’information dans le back-end. Peu avant l’aboutissement de cette année de licence professionnelle web, j’ai eu l’idée d’un projet informatique qui me permettrait de réaliser tous les autres plus facilement. J’ai donc, le 27 octobre 2019, commencé un prototype d’environnement de développement intégré (IDE).

Après obtention de mon diplôme de niveau Bac + 3, j’ai été embauché le 25 décembre 2019 par l’agence web clermontoise De Bussac Multimédia, en tant que développeur front-end. J’y ai travaillé pendant 2 ans et 6 mois, finissant Lead Dev sur un projet React important et complexe. Ce premier CDI m’a permis d’apprendre à maîtriser des compétences qui ne nous avaient pas été enseignées lors de mon cursus plus orienté back-end. Les dates limites et l’exigence du rendu étaient très importantes, j’ai dû améliorer ma productivité pour arriver à mes objectifs. Sans cette expérience je n’aurais pas connu l’École Hexagone, ni eu la volonté et la patience nécessaire à l’aboutissement de cette fin d’étude réalisée en alternance chez ABGX, de nouveau

sur un poste de développeur full-stack. J'ai ainsi pu travailler sur une solution logicielle de gestion de la radioprotection tout en continuant mon éternel apprentissage et obtenant ainsi un Bac + 5 avec la réalisation de ce mémoire. Depuis le 27 octobre 2019 et durant toutes ses années, ma détermination à développer un IDE convivial et intuitif n'a cessé de grandir. Je sais d'ores et déjà que je vais prochainement créer une entreprise et réaliser de la prestation pour financer la réalisation de ce projet. Le problème auquel il répond est celui de ce mémoire.

Remerciements

Je ne remercierai jamais assez mes proches, amis, et membres de ma famille, notamment mes parents et mes grands-parents paternels, propriétaires d'ordinateurs et de connexions internet avant ma naissance en 1999. Grâce à cela, ils ont eu la démarche raisonnée de m'introduire à cette technologie, sur des logiciels éducatifs qui m'ont appris à lire à l'âge de 2 ans. Merci également à mes grands-parents maternels qui m'ont transmis les bases de la musique et du chant grâce à leur dévotion à la chorale et à l'orchestre des messes de Rouffignac.

Merci à toutes les personnes des entreprises et écoles qui m'ont accueilli, lors de ma scolarité et de mon début de carrière, me transmettant une majeure partie de ce que je sais aujourd'hui, et ce dans la bienveillance, grâce à des processus itératifs d'amélioration continue.

Merci tout particulièrement à l'*École Hexagone* et à *ABGX* sans lesquels je n'aurais sans doute pas pris le temps d'écrire ce mémoire et donc ni découvert toutes les informations qu'il contient, ni affûter mon esprit, ma volonté, et ma détermination.

Merci aux logiciels libres, à l'open source, à Wikipédia et tous leurs contributeurs.

Merci à toutes et tous, car nos métiers et découvertes ne sont que rarement le fruit d'un génie isolé, mais bien d'une collaboration et de l'amélioration d'idées antérieures.

Enfin, merci à toute personne qui lit actuellement ce texte. Demandez vous pour qui vous le faites, la réponse devrait toujours être $\forall (soi \wedge autrui)$. La réponse est mathématique et logique. Si elle ne vous est pas évidente, je vous donnerai de quoi déchiffrer cette syntaxe.²

Introduction

La transmission d'informations s'est fiabilisée au fil des époques, avec l'usage d'outils physique ; notamment des tablettes, plaques permettant de dessiner, d'écrire, lire, compter ; et ce jusqu'à pouvoir automatiser son traitement et son stockage, à l'aide de l'informatique, dont l'histoire ne commence concrètement qu'à la fin du 19^{ème} siècle. De nos jours, nous interagissons quotidiennement avec des ordinateurs, que ce soit à l'aide de la programmation, ou des interfaces hommes-machines qui en sont issues. Pourtant, la grande majorité des utilisateurs, et même des personnes travaillant dans le domaine des Technologies de l'Information et de la Communication, ne connaissent pas la plupart des événements majeurs de son histoire. Moi-même n'ayant pas cette connaissance lorsque j'ai entrepris la rédaction de ce mémoire, j'ai décidé de me documenter et de rédiger un format accessible, avec ce qui m'a semblé être nécessaire pour comprendre le sujet, et ainsi le partager à vous, lecteurs de cet ouvrage. Pour la rédaction de ce dernier, j'ai donc acquis trois livres sur l'histoire de l'informatique qui ont chacun leurs avantages grâce à une approche différente d'un même sujet, sur lequel je m'attendais à trouver plus de bibliographies. Contrairement à ceux-ci, qui apportent une vision globale de l'évolution de l'informatique et des machines, j'ai abordé une démarche axée sur la gestion de l'information dans le sens large qui finira par se concentrer sur la programmation, le tout avec des pointes de philosophie, ainsi que mon humble analyse subjective, car j'aime penser. L'apprentissage des mathématiques et de l'informatique ont changé ma manière de concevoir le monde dans lequel nous vivons. Ce sont selon moi des philosophies à part entière, apportant des syntaxes et un vocabulaire, nécessitant un apprentissage constant de schémas de pensées dignes d'un logicien.

Dans ce mémoire, je vais donc revenir aux racines les plus ancestrales de la gestion de l'information, au vocabulaire et champ lexical qui y sont liés ou en découlent, ainsi qu'à leurs étymologies à travers leur [Histoire](#), qui représente toute la première partie de ce livre. J'y présenterai quelques philosophes et penseurs, mais surtout des scientifiques variés. De l'Antiquité aux époques plus modernes, les idées qu'ils ont amenées ont permis des inventions et progrès technologiques avec un développement de plus en plus rapide. Pourtant, les inventions qui ont le plus impacté l'humanité resteront toujours les plus anciennes, avec pour origine l'écriture et surtout l'art. Très tôt, l'humanité s'est intéressée au ciel, dont l'immensité lointaine nous est répétitive et prévisible. Si je parle d'astronomie dans ce mémoire, c'est que le plus ancien calculateur analogique retrouvé servait à prédire la position ou la date de venue des astres grâce à des calculs astronomiques complexes. Les machines et automates sont depuis l'antiquité utilisés pour nous faciliter la vie. Après avoir présenté les origines de l'informatique et des mécanismes logiques, j'aborderai un sujet connexe qui m'est cher, la programmation. Elle n'est pas forcément liée à l'informatique. C'est l'art de planifier et mettre en œuvre des étapes à suivre pour obtenir un résultat. Cela dit, elle passe inévitablement

par de la gestion de l'information, à l'aide d'écriture, lecture, chiffrement, déchiffrement, émission, et réception. Je m'intéresserai en outre à l'apprentissage, qui représente l'acquisition et l'assimilation d'une information ou d'un système. Cette notion est connexe à l'intelligence artificielle ; ainsi qu'à la pédagogie, propre à « l'animal social émotionnel » qu'est l'humain ; et à la compréhension du rapport maître-élève en tant que canal de la transmission de compétences. C'est à l'aide de l'étude de tous ses sujets liés à l'information et à l'automatisation, que je vais repenser la gestion de l'information et moderniser l'expérience développeur.

L'ensemble de ces domaines, et l'information en général, prennent une place de plus en plus importante dans nos quotidiens. Cela s'accompagne d'un volume exponentiel de données mondialement échangées. Nous avons développé une relation symbiotique avec la technologie, qui nous permet aujourd'hui de communiquer et de jouer presque instantanément au niveau mondial, améliorant globalement les conditions de vie humaines. Pour autant, comme toutes les technologies récentes ayant un impact majeur, viennent un rejet et une technophobie d'une certaine partie de la population pas encore convertie à son usage. Avec mon point de vue de développeur web, je dresserai ensuite un bilan de l'état actuel de l'informatique, de son marché, de ses opportunités, ainsi qu'une étude de sa pratique moderne ; et ce afin d'en identifier les problèmes que nous y rencontrons pour en dégager de possibles solutions. En tant qu'artisan du code, j'aboutirais ainsi ce mémoire avec la mise en place technique d'une solution informatique, pour ensuite finir par présenter ma vision pour le futur de cette discipline, plus particulièrement la pratique que je souhaite en avoir en tant que créateur de services et contenus web.

Comprendre la problématique

L'informatique, étymologiquement « automatisation de la gestion de l'information », est un domaine récent et complexe, qui soulève beaucoup d'avancées technologiques, mais également de craintes et de questionnements. En 2022, la programmation à l'origine des logiciels qui nous permettent d'utiliser le matériel informatique n'était pratiquée que par 0,35% de la population. Elle requiert une rigueur suffisante pour dans un premier temps passer la compilation si elle a lieu, puis l'exécution du programme en résultant, et enfin les tests du comportement voulu dont le but est de vérifier que nous avons correctement implémenté la solution à notre problème informationnel. Une fois que tout cela est satisfait, il faut rendre le programme robuste afin d'assurer sa pérennité tout au long du développement des nouvelles fonctionnalités qui s'accumuleront inexorablement, et ce afin que toutes les existantes restent fonctionnelles. C'est ce que l'intégration et le développement continu solutionnent, en assurant la qualité d'un logiciel au fil de son développement. Dans la deuxième partie de ce mémoire, je montrerai ainsi les avantages et inconvénients des outils de développement actuels et les biais introduits dans leurs pratiques, constituée des étapes décrites précédemment, avec des solutions.

Actuellement, l'apprentissage de ces étapes et les coûts de formation d'un nouvel employé en informatique sont importants. Simplifier la prise en main d'un projet informatique me semble être une opportunité technologique intéressante, qui permettrait à l'industrie informatique de se concentrer sur la création et l'innovation. Dans cette dernière, la matière première est la pensée du développeur, c'est l'acteur principal à l'origine des algorithmes qui régissent la gestion de l'information.

Nous réalisons des logiciels pour de multiples industries et sommes pourtant les cordonniers les plus mal chaussés. Nous parlons souvent d'expérience utilisateur, mais l'expérience développeur reste à désirer, voire archaïque. Nous sommes temporellement proches de la genèse du logiciel et des interfaces utilisateurs. La question fondamentale à laquelle je répondrais dans le troisième chapitre sera donc « *Comment repenser la gestion de l'information pour moderniser l'expérience développeur ?* ». Avant d'y apporter une solution technique, il est important de comprendre l'évolution de la pensée, des outils, des machines et de la technologie, pour savoir comment l'humanité en est arrivée là. Comment elle calculait, automatisait des actions, et gérait l'information avant l'invention des ordinateurs ?



F2 : meme sur l'expérience développeur

Histoire

Pour tout être vivant, la première manière de marquer l'Histoire est universelle, c'est celle des fossiles et des dinosaures, celle d'avoir existé et laissé une trace, une information, stockée sur un support physique, quelque part sur sa planète. La deuxième est l'art de sculpter des outils, actuellement estimé 3 millions d'années avant notre ère. La troisième est le dessin, au travers de la peinture ou de la gravure, dont les plus anciennes preuves sont rupestres, et âgées de 65 millénaires. Relativement aux dates précédentes, l'écriture ne vient que récemment, elle a été attestée en Mésopotamie il y a 5 millénaires. Pourtant, c'est elle qui nous permettra à nous, humains, d'écrire notre Histoire, de transmettre des volumes d'informations variées, véridiques comme les faits, ou de qualité comme des idées. Écrire permet de faire naître une information et potentiellement de la rendre éternelle. Les paroles s'envolent, les écrits restent, c'est pourquoi la littérature est importante. D'un point de vue historique, notre génération laissera toujours une trace. Que ce soit sur un disque dur quelque part dans un grenier, dans un datacenter ou une archive internet. Et même si tous les humains disparaissaient, il y en aurait toujours une trace immuable et indestructible à échelle de la Terre. Les isotopes lourds que nous avons créés avec l'apparition des industries nucléaires seraient, même dans le cas d'un retour à l'âge de pierre causant un rude hiver civilisationnel, une preuve de l'existence de la nôtre.

Préhistoire

Avant même la naissance de l'écriture qui marque la fin de la préhistoire, les humains avaient déjà besoin d'informations quantitatives, c'est-à-dire de compter et d'effectuer des calculs. Le premier réflexe a sûrement été de compter sur ses doigts, raison pour laquelle la base 10 est si présente dans l'Histoire. Pour des nombres plus importants, il a fallu inventer d'autres stratagèmes et outils. Un **abaque** est un instrument facilitant le calcul. C'est un outil mnémotechnique de numération, c'est-à-dire qu'il permet d'enregistrer un nombre pour se libérer de la mémoire. Son étymologie est un mélange de grec et d'hébreu qui explique bien sa représentation et son usage. La poussière (de l'hébreu abaq) et la tablette (du grec akos) servaient de support pour l'écrit. Même sans dépôt de poussière naturelle, il est possible de volontairement recouvrir une tablette, et ainsi réécrire à volonté avec du sable, ou toute autre poudre. À la préhistoire, le mot abaque n'existait pas, les humains n'avaient pas de tablettes, de nombres ni de textes leur permettant de représenter une quantité. Ils ont pourtant inventé le plus vieux système de quantification connu. Le **bâton de comptage**, daté à **-40000**, est un système unaire. C'est-à-dire qu'il permet de quantifier des unités, représentées par la répétition d'un symbole, généralement un trait, aussi appelé marque de dénombrement. Le plus ancien est l'Os de Lebombo, mais l'ensemble d'Os d'Ishango est cependant plus connu. Il serait hypothétiquement une des premières preuves de connaissance humaine des mathématiques et de l'arithmétique. Sur un des os, les 4 nombres premiers entre 10 et 20 sont présents, soit 11,

13, 17, et 19. Un autre semble démontrer les additions et les multiplications, notamment la duplication, avec la juxtaposition des nombres 3 et 6 puis 4 et 8. Cette méthode unaire est longtemps restée, à l'époque, un berger pouvait s'assurer que l'intégralité de son troupeau était rentrée en comparant deux comptages, à l'aide de cailloux par exemple, chacun représentant une bête. J'ai récemment eu l'opportunité de visiter la grotte de Thaïs, où a été retrouvé un os de quelques centimètres aux multiples gravures. Daté en **-12 500**, il a été nommé **Os coché**. En 1991, l'archéologue américain Alexander Marschack démontra qu'il ne s'agit pas d'une représentation décorative comme les historiens le pensaient alors, mais d'un système d'enregistrement du temps, possiblement un calendrier basé sur des observations astronomiques. Cette hypothèse est actuellement, en 2024, le sujet d'une étude européenne menée par un groupe de chercheurs à l'université de Bordeaux.

L'homme préhistorique avait donc sûrement déjà des systèmes de comptage des unités et du temps, bien qu'il ne nous en reste que peu de traces, ne permettant donc pas d'affirmer cela avec certitude. À cette période, l'humain était un chasseur-cueilleur nomade, et contrairement à ce que nous pourrions croire, il aurait rarement été hostile à ses congénères lorsqu'il les croise, ayant déjà bien assez de problèmes avec la nature. La **sédentarisation** prend place en **-9000**, l'**agriculture** apparaît à la même période dans le *croissant fertile* situé au Proche et Moyen Orient actuel, avec par conséquent un besoin de stockage grandissant auquel répondra la démocratisation de la vannerie, du tissage et surtout de la poterie. À partir de **-7000** fleurissent alors des prémices de **villes et cités** comme Çatal Höyük et Ninive. Suivies par Uruk et Lagash vers -5000, puis Ur vers -4000. Les **premières civilisations** comme celle de Sumer sont apparues vers **-3500**, celle de la vallée de l'Indus, à qui l'humanité doit entre autres les égouts, est quant à elle apparue vers -2600, bien que son développement ait commencé à l'apparition des premières villes et cités. En découlent Assur, capitale assyrienne, et Guzana, capitale du royaume antagoniste araméen vers -1500. L'arrivée du stockage de biens a causé l'apparition des premiers conflits d'envergure. Avant cela, les scientifiques estiment que les humains avaient une densité de population trop faible pour que cela n'arrive, le rapport gain / perte n'en valant que rarement le coût. L'humanité a donc dû, suite à la sédentarisation, trouver des moyens de réguler son jeu de la vie humaine, et trouver des systèmes composés de règles, permettant de conserver l'ordre et éviter la discorde.

C'est ici qu'intervient la **bulle enveloppe V comptable**. Il y a presque 6000 ans de cela, en Mésopotamie, vers l'an **-3900**, étaient utilisées des sphères d'argile. Des calculs, du latin « calculus », signifiant « petits cailloux », étaient stockés en son sein pour en comptabiliser le contenu. L'extérieur de ses sphère d'argiles comportait des reliefs représentant les dieux ou le pouvoir. Cette empreinte était réalisée avant formation de la sphère, en déroulant un sceau-cylindre sur une plaque d'argile fraîche. Ils servaient à imprimer les motifs creusés dessus. Certains étaient composés de matériaux précieux, et les plus récents comportaient parfois des écritures cunéiformes. Cela indiquait généralement le possesseur à la manière d'une signature.

Les Mésopotamiens avaient donc inventé un système de gestion de l'information permettant de garantir le transport et l'échange de marchandises par un transporteur, tout en garantissant des informations comme sa provenance et la quantité de marchandise. En effet, un destinataire pouvait ainsi faire confiance à un transporteur concernant la provenance, grâce à la signature ; et la quantité, en comparant le nombre de marchandises au nombre d'unités présentes dans la bulle après l'avoir cassé. Si elle l'était avant d'arriver à son destinataire, alors la livraison pouvait être invalidée.

Antiquité

Mésopotamie

Marquée par l'invention la plus importante de notre histoire, la naissance de **l'écriture** datée vers l'an **-3250** a permis une transmission d'informations complexes sur un support théoriquement permanent. Des lois, qu'elles soient juridiques, mathématiques, ou autres, ont alors pu être rédigées et transmises de manière plus fiable que par la transmission orale, pratiquée jusque là en plus du dessin. Malgré cela, beaucoup de penseurs ont continué à transmettre leurs savoirs exclusivement à l'oral, n'ayant donc jamais rien écrit de leur vivant. Bien que des gens l'ont fait pour eux, comment être sûrs que les informations de leurs discours n'ont pas été déformées, par des siècles de copies et réécriture, de ragots et légendes. Pour autant, ce qu'il en reste est une facette de la réalité, c'est ce qui a persisté jusque là grâce à ce que les auteurs ont jugé important de transmettre aux générations futures.

Dès l'Antiquité, les Égyptiens et les Babyloniens avaient déjà rédigé des textes comportant des problèmes, et des méthodes mathématiques pour les solutionner, tels que le Papyrus Rhind, qui explicite notamment des opérations comme la division, la multiplication, et bien d'autres. **L'algèbre babylonienne** datée vers **-2600**, précède le Papyrus Rhind d'un millénaire et utilisait un système sexagésimal. Ce système, potentiellement emprunté aux Sumériens, était de base 10, fondation du système décimal ; et 60, toujours utilisée de nos jours pour le temps, ou les angles. Cette algèbre possédait bien plus de distinctions d'opérations logiques que les simples additions, multiplications et divisions. Certains concepts peuvent aujourd'hui se retrouver en programmation. Ils avaient deux manières d'additionner. L'empilement qui donne un troisième nombre à partir de deux autres, soit $a = b + c$; et l'ajout par fusion d'un autre nombre à un premier avec $a += b$, qui équivaut à l'empilement $a = a + b$. Idem pour la soustraction avec $a = b - c$, ou $a -= b$ analogue à $a = a - b$. Avec de surcroît la notion de comparaison et d'équation. Si après $a - b$ il reste c , alors $a > b$ car $a = b + c$.³

Les usages de la multiplication sont eux-mêmes multiples, ils avaient déjà des méthodes permettant de répéter un nombre, ou de calculer une aire comme un volume. Afin d'avoir le résultat d'un calcul plus rapidement, les mathématiciens du début de l'antiquité ont tout simplement calculé toutes les combinaisons de multiplications entre les chiffres. Créant ainsi ce que toute personne introduite aux mathématiques a dû apprendre par cœur un jour, **les tables de**

multiplications, comparables à un cache informatique permettant de ne pas refaire le calcul pour obtenir un résultat (complexité O^1). Les historiens pensent que la division se pratiquait avec la multiplication par l'inverse, car les archéologues en ont retrouvé de multiples tables dans la Mésopotamie. La racine carrée était obtenue grâce aux tables de carrés, et pour les valeurs manquantes, les mathématiciens de l'époque devaient pratiquer une approximation par interpolation linéaire, c'est-à-dire une estimation de la valeur sur une courbe entre deux points connus. Cette discipline qu'est l'Algèbre est donc à ce moment de l'histoire avérée, mais pas encore nommée, du moins pas en tant que telle, il faudra attendre le Moyen-Âge pour cela.

Le **quipou** est un abaque inca qui signifie « nœud », ou « compte » en quechua. Il est en effet possible de faire un nœud simple avec plusieurs boucles, permettant ainsi d'enregistrer un chiffre correspondant au nombre de boucles, et donc des nombres, en base 10. Le plus ancien connu a été retrouvé sur le site archéologique de Caral, au Pérou. Il est daté vers l'an **-2500**. Le climat chaud et sec a contribué à sa bonne conservation. L'Empire inca ayant été constitué de plus de 12 millions d'individus, cet outil a été au cœur de son administration et a permis le stockage des informations nécessaires au recensement de la population et à la gestion de l'économie. Des experts du quipou, dont Gary Urton⁴, professeur d'Anthropologie américain spécialisé en civilisation précolombienne, se sont même rendu compte que plus d'un tiers comportaient des informations autres que des chiffres, aussi appelées informations qualitatives, en opposition aux informations quantitatives. Cet outil a donc de surcroît servi à transmettre des messages, composés d'un vocabulaire créé à partir de différentes couleurs, types de fils, de nœuds, ainsi que leurs positions et orientations. Notez qu'en mathématique il existe une branche très intéressante appelée *théorie des nœuds*, elle fait partie de la sous-branche de la *topologie*, c'est-à-dire l'étude de la déformation d'un objet sans arrachage ni recollage.


Le quipou nous montre qu'à travers la manipulation des fils et des nœuds, certaines civilisations ont pu écrire et lire des informations, autrement qu'en inscrivant des symboles sur une surface. Le dernier abaque antique est le boulier, qui a été utilisé par de nombreux peuples séparés. Il voit le jour possiblement vers l'an **-2000**, et utilise le système de numération décimale (10), ou en base alternée (5, 2) pour certains modèles. C'est un outil physique composé de plusieurs rangées d'unités sphériques, des perles empilées glissant autour de tiges, permettant un calcul rapide, même récemment. Bien utilisé, il arrive en tête de compétitions contre des opérateurs électroniques. Cet outil répandu a été démocratisé dans de multiples peuples. Romains, chinois, japonais, russes, mexicains, français, et bien d'autres, l'utilisent et l'enseignent.

Le Code de Hammurabi, un texte babylonien daté vers **-1750**, est à ce jour le texte de loi connu le plus complet de la Mésopotamie Antique. Il démontre l'existence de lois, notamment concernant les salaires, indiquant qu'il existait des corps de métiers bien définis et inscrivant entre autres le droit de la famille, de la propriété, ainsi que le système judiciaire. Ces lois décrivent des situations problématiques auxquelles elles proposent des solutions, elles sont *casuistique*. À l'époque où Hammurabi succède à son père, il possède un petit territoire

comportant tout de même de grandes villes, mais entouré de puissants royaumes qu'il finira par vaincre et annexer, pour finir par dominer la région et fonder le royaume babylonien tel qu'il est connu en tant que puissance culte du Proche-Orient. La politique rédigée à sa gloire dans ce code éponyme a donc certainement eu un rôle dans le développement de la société babylonienne jusqu'à son apogée.

Grèce

Les premiers écrits philosophiques et scientifiques de référence occidentale viennent de la **Grèce antique**. Les bases de la physique ont été posées par Thalès, celles des mathématiques par Pythagore, de la géométrie par Euclide, et de la biologie par **Aristote**, qui est à l'origine des raisonnements logiques appelés **sylogismes**, étymologiquement ensemble de discours logiques. Ce sont des raisonnements constitués d'au moins trois propositions cohérentes, la dernière étant une conclusion déduite, précédée de prémisses qui doivent s'avérer vraies pour que la conclusion le soit également. Soit $(A \wedge B \Rightarrow C)$. Son exemple le plus connu est « *Tous les hommes sont mortels, or Socrate est un homme, donc Socrate est mortel* ». $(H \Leftrightarrow M \wedge S \Leftrightarrow H \Rightarrow S \Leftrightarrow M)$ Socrate est la première figure philosophique dont l'influence a marqué un avant et un après, il a été enrôlé en tant qu'hoplite lors de la guerre du Péloponnèse qui a opposé Sparte à Thèbes et Athènes. Dans le cadre de cet affrontement, les humains ont eu besoin de communiquer des messages à de longues distances, sans qu'ils se fassent intercepter, et encore moins décoder et lus, voire pire, compris. Le plus ancien système de **chiffrement** connu est ainsi la **scytale** spartiate, datée vers **-600**. Utilisée pour transmettre des messages chiffrés par transposition sur une bande, généralement une ceinture en cuir portée pendant le transport, à enrouler autour d'un bâton pour l'écriture et la lecture. Le diamètre du bâton étant ainsi la clé de chiffrement, deux bâtons de mêmes diamètres pouvaient donc chiffrer et déchiffrer les mêmes messages. Les lettres correspondaient à leur propre valeur, et leur ordre dans le mot était conservé, malgré la présence de lettres entre chacune d'entre elles. Ces dernières étaient simplement mélangées, et l'enroulage permettait de les réaligner afin d'en permettre la lecture. Quelques années avant Jésus Christ, Jules César encodait ses messages grâce à un procédé similaire en utilisant un code éponyme. Ce chiffrement simple utilisait un décalage d'un certain nombre de lettres de l'alphabet. A vaut D, B vaut E, Z vaut C, etc. Remarquez que le chiffrement était dès le début utilisé pour transmettre des informations sensibles, notamment liées à la guerre. Cet outil a d'ailleurs été considéré comme arme de guerre jusqu'à récemment, où la plupart des techniques de cryptographie issues du domaine militaire sont devenues publiques. Malgré cela, les gouvernements ont récemment appelé à retirer l'anonymat sur internet ou à avoir des portes dérobées dans les chiffrements afin d'en permettre la lecture par une autorité certifiée en cas de besoin.

La transmission de messages, le chiffrement, la programmation et l'informatique reposent tous sur les **mathématiques**. Étymologiquement, ce mot signifie en latin « qui aime apprendre », ou en grec, « qui provient d'une leçon », autrement dit : du savoir d'autrui. Sa définition moderne, présente dans le Larousse, comme sur Wikipédia, vient corroborer cette étymologie, indiquant que c'est un ensemble de connaissances abstraites résultant de raisonnements **logiques**, appliqués à des objets tangibles, tels que les formes physiques, les structures qu'elles constituent, et leurs transformations possibles à l'aide des relations existantes entre ces objets. Ces connaissances abstraites, ce sont les nombres, ainsi que les opérations, formules et théorèmes qui nous permettent de les utiliser. Cette abstraction mathématique basée initialement sur des axiomes tangibles, c'est-à-dire des règles admises comme vraies sans démonstration, s'est petit à petit détachée de toute contrainte physique, et a ainsi donné naissance à deux disciplines, les mathématiques *appliquées*, et les mathématiques *pures*. 400 ans avant J.-C., Platon a contribué à cette distinction, séparant la technique de calcul ; « *appropriée pour l'homme d'affaire et de guerre* », qui doit quantifier et gérer ses troupes, gains et pertes ; de la théorie des nombres « *nécessaire au philosophe pour surplomber la mer des changements et s'emparer de ce qui est véritable* ». Les **mathématiques pures** tendent à la généralité, ou en langage de mathématicien, vers la factorisation, c'est-à-dire la simplification d'un concept sous une forme plus courte et facile à comprendre, donc à transmettre et réutiliser. Qui plus est, notez que c'est une notion très présente en programmation, la refactorisation. Elle fait partie des bonnes habitudes et de l'amélioration continue. Les **mathématiques appliquées**, quant à elles, utilisent ces concepts sur des domaines très spécifiques, pour exemple, la **géométrie**, étymologiquement « science de la mesure du terrain », a été développée dans l'Égypte antique pendant les trois siècles avant J.-C., à partir du besoin de mesurer la superficie d'un champ, aussi appelé arpentage. En définitive, cette discipline que sont les mathématiques est une philosophie en soi, c'est-à-dire un système d'idées permettant de conceptualiser et visualiser le monde tout en le quantifiant. Même étymologiquement, il y a une proximité entre les deux mots pour le savoir, par le partage de connaissance, dispensé lors d'une leçon d'autrui. Entre temps apparaît la **sténographie**, du grec ancien « écriture courte », c'est un procédé de tachygraphie, autrement dit d'écriture rapide. Ce processus de réécriture syntaxique trouve ses plus vieilles traces connues à la même période. Le « langage SMS » a ainsi été avéré grâce au bibliographe et doxographe Diogène Laërce qui a documenté beaucoup d'informations sur les personnalités grecques antiques. Les premières traces d'abréviations remontent ainsi à **405 avant J.-C.**, date à laquelle Xénophon, élève de Socrate, a usé de sténographie pour transcrire les discours de son maître, face au besoin d'écrire rapidement et idéalement à la vitesse de la parole. 

Égypte

En **-336**, un disciple d'Aristote connu sous le nom d'**Alexandre le Grand** hérite alors, à l'instar d'Hammurabi, du puissant royaume macédonien avec sa forte armée qui lui permit de devenir l'un des plus grands conquérants. Il s'empara de l'Empire perse et arriva jusqu'en Égypte où il fut proclamé pharaon. Véritable Roi-bâtitteur, il a fondé une vingtaine de cités nommées **Alexandrie**, dont la plus connue et importante sera celle d'Égypte, fondée en **-331**. Il décédera en **-323**, et l'humanité assiste alors à une transmission des informations de la culture grecque en Égypte. La mort d'Alexandre le Grand marque le début de la période hellénistique, des Mouséïons apparaissent alors dans toute l'Égypte antique. Ce sont des sanctuaires consacrés aux muses, déesses des arts. Celle d'Alexandrie abritera la célèbre **bibliothèque** éponyme fondée vers **-288**. Elle réunissait à son apogée environ 400 000 des plus importants ouvrages antiques, sous la forme de rouleaux de papyrus. Bien qu'une partie des œuvres ait été achetée, Ptolémée Ier, qui succède alors à Alexandre le Grand, aurait demandé à tous les navires arrivant à Alexandrie de transmettre les ouvrages qu'ils transportaient pour recopie, gardant l'original à la bibliothèque d'Alexandrie. Ce mode d'acquisition de l'information est l'ancêtre du dépôt légal incitant ou obligeant les producteurs et diffuseurs à déposer leurs œuvres pour les compiler dans le bien commun. Par la suite Ptolémée II demanda aux rois et grands de ce monde de transmettre leurs œuvres. Enfin, Ptolémée V interdira l'exportation d'œuvres afin que la bibliothèque d'Alexandrie en ait l'exclusivité. C'est une preuve flagrante que l'acquisition ou la rétention d'information est un pouvoir à part entière.

Le troisième siècle avant J.-C. est celui des origines de la logique mécanique. Les **mécanismes** en résultant sont en effet des assemblages d'éléments constituant des processus logiques. Une force en entrée provoque une chaîne d'actions mécaniques qui facilite voire automatise le résultat en sortie. **Archimède** de Syracuse naquit en **-287**, il fut élève d'Euclide avant d'être considéré comme le plus grand mathématicien de l'Antiquité. Il a beaucoup apporté à la géométrie et à la mécanique, pour lesquelles il a étudié et rédigé de multiples traités. Il a entre autres rédigé le fameux théorème de la poussée d'Archimède et l'explication du principe du levier, respectivement à l'origine du « Eurêka » et du fameux « Donnez-moi un levier, un point fixe, et je soulèverai la Terre ». Il est l'inventeur de mécanismes comme le palan qui permet de soulever des charges lourdes à l'aide d'un double système de poulies. Moins connu, mais pas des moindres, **Ctésibios** est un ingénieur né en **-284** à Alexandrie, où il aurait fondé l'école des mécaniciens. Il s'intéressa dès son plus jeune âge à la mécanique et aux machines hydrauliques. À seulement seize ans, il inventa un monte-charge hydraulique. Par la suite il a perfectionné la clepsydre en ajoutant un réservoir intermédiaire maintenu à un niveau constant, permettant un flux qui l'est également et offrant ainsi une mesure du temps plus précise. S'ajoute à cela l'invention du premier orgue de l'histoire, à l'aide des pistons, soupapes et **claviers** dont il est aussi à l'origine, à une époque où les moyens techniques sont pourtant limités.

Né en **-280**, **Philon de Byzance** est le plus ancien mécanicien dont les historiens aient retrouvé la majorité de ses œuvres, qui seront utilisées de l'Empire romain au 10^{ème} siècle. Il rédigea moult traités sur les leviers, pneumatiques, automates, clepsydres, constructions, roues et machines de guerre. Il a documenté beaucoup de choses dans ces domaines et c'est grâce à lui que l'humanité connaît les réalisations de Ctésibios. Il fut aussi inventeur, ou du moins il est le premier à décrire le thermoscope, permettant de visualiser les différences de températures. De plus, il réalisa des **automates** sonores et des systèmes d'approvisionnement automatique de liquides, dont un automate de servante, permettant à l'utilisateur d'y déposer un verre, déclenchant un système faisant s'écouler du vin puis de l'eau, avec la possibilité de retirer son verre avant ou pendant l'écoulement de l'un ou de l'autre pour arrêter le mécanisme.

Peu après cette époque où fleurissent les premiers mécanismes d'automatisations, les archéologues ont retrouvé ce qui est actuellement le plus ancien calculateur analogique attesté. **L'Anticythère**, daté vers **-150**, est le plus vieux mécanisme à engrenage connu. Réalisé en bronze, il servait à prédire les éclipses solaires et lunaires. Il est constitué d'un cadran de 233 positions correspondant au nombre de mois espaçant deux éclipses, soit un cycle nommé Saros ; d'un cadran métonique pour indiquer le mois et l'année ; et d'un cadran de 365 positions, correspondant au nombre de jours d'une année civile du calendrier égyptien, décrit dans le Papyrus Rhind. Ainsi, en actionnant les engrenages à l'aide d'une potentielle manivelle non retrouvée sur l'Anticythère, l'utilisateur pouvait retrouver les différentes dates des éclipses. L'existence d'un artefact aussi unique et complexe que celui-ci reste un mystère pour les historiens. À cette même période, se développent les **routes de la soie**, une des voies de transport et d'échange de marchandises les plus importantes de l'Histoire. À partir de **-130**, ce réseau passant par l'Europe, le Moyen-Orient, l'Asie et l'Afrique de l'Est, permet des échanges culturels et scientifiques.

Marcus Vitruvius Pollio, connu sous le nom de **Vitruve**, est un architecte romain qui a œuvré pendant le **1er siècle avant J.-C.** Il a étudié et relaté les travaux architecturaux de l'Égypte antique, notamment la construction d'Alexandrie par Dinocrate⁶, compilant les travaux des savants de cette période. Bien qu'il ne soit pas de l'Égypte Antique, sans lui l'humanité aurait perdu beaucoup d'informations de cette époque. À travers ses recherches et réalisations, il a œuvré à l'essor de la Rome Antique⁷. Héritant des savoirs d'Archimède, il nous a transmis des informations liées à ce dernier. De même, il a décrit bon nombre de machines utilisées de son temps pour la construction, mais aussi pour la guerre, comme la catapulte, la baliste, et bien d'autres. D'un point de vue de l'information, Vitruve a réalisé un travail de recherche et de retransmission très efficace dans le domaine de la construction liée à l'eau, notamment sur les aqueducs et les siphons. Son livre « *De architectura* » est le seul qui nous reste sur l'architecture de l'Antiquité classique. Malgré ses quelques innovations, il a surtout posé les codes de l'architecture, devant être solide, utile et belle. Les six principes théoriques qui la régissent sont l'ordonnance, la disposition, l'harmonie, la symétrie, la convenance, et la distribution. En effet, pour mener de telles réalisations architecturales à bien, il faut que la

création soit correctement disposée dans son environnement. Et ce afin d'avoir une certaine harmonie et une conception symétrique, autant par praticité que pour la beauté imitant la nature. Enfin, l'ordonnance, la convenance, et la distribution ; sont nécessaires pour gérer les équipes travaillant sur un projet, afin que tout le monde trouve son compte dans sa réalisation.

Originaire d'Alexandrie, **Héron** serait né et aurait vécu pendant le **premier siècle après J.-C.** Cet ingénieur a utilisé les principes mécaniques pour automatiser certaines tâches grâce à des automates, notamment pour des scènes de théâtres. Certains historiens lui attribuent la création de la première machine à vapeur, l'Éolipyle, qui servait uniquement à distraire ses utilisateurs, car l'énergie dégagée était négligeable. Pour autant, Vitruve aurait déjà eu mentionné de telles machines à l'époque, et il s'est avéré par la suite qu'Héron était postérieur à ce dernier, qui décéda en -20. Il a donc juste démocratisé des machines à eau et à vapeur, qu'il aurait, selon ses explications dans son livre *Pneumatica*, perfectionné au point de pouvoir ouvrir les portes d'un temple à la force de la vapeur, obtenue en chauffant de l'eau à l'aide du feu d'un autel.

Pour clôturer cette partie sur l'Antiquité, je vous présente **Claude Ptolémée**, un astronome, mathématicien, et géographe grec qui a vécu à Alexandrie. Tout comme Aristote, il pensait que nous étions dans un système géocentrique, où les planètes et le Soleil tourneraient autour de la Terre. Malgré cela, son traité d'astronomie « *L'Almageste* » est le seul ouvrage complet connu de l'Antiquité, il a décliné cette œuvre en rédigeant « *les tables faciles* ». Très utilisées à l'époque, elles permettaient de calculer la position des astres et les éclipses avec une précision alors suffisante. Il faudra attendre [Copernic](#) pour que ses travaux soient remis en question et ne fassent plus référence.

Moyen-Âge

Les civilisations antiques ont alors introduit le calcul et la gestion de l'information à travers l'écrit de multiples textes, certains se perdant au fil du temps, d'autres étant conservés grâce à des personnalités qui les ont repris et fait vivre en les appliquant voire les améliorants. Les différents penseurs de l'Antiquité ont donc défini, catégorisé et classifié les informations de la connaissance humaine en modules. Ce concept omniprésent a permis une standardisation des idées et une base permettant d'avoir un langage commun. À l'aube du Moyen-Âge, les savants ont alors découvert et transmis les règles fondamentales des mathématiques, de la biologie, de la physique, et surtout de la mécanique permettant l'automatisation de l'agriculture par un meilleur approvisionnement en eau, ou encore la création d'automates et de machines de guerre. En -47, [Jules César](#) ordonne d'incendier la flotte d'Alexandrie, détruisant une grande quantité de richesses et potentiellement de savoirs, contenue dans la bibliothèque ou dans ses alentours. Sous la Rome Antique, les maîtres artisans n'enseignaient qu'à leurs apprentis pour minimiser la surface d'attaque sur leurs systèmes de connaissances, et ainsi avoir de la renommée. C'est pour cela que les textes de Vitruve ont été si précieux pour les historiens de cette période. Et c'est potentiellement pour une raison antérieure semblable, que l'Anticythère est le seul mécanisme antique de précision à avoir été retrouvé par les archéologues.

Bien que le début du Moyen-Âge soit marqué par un déclin avec la **chute de l'Empire romain** en **450**, il se passera bien des choses dans le domaine des mathématiques, de la cryptographie et de l'imprimerie. **Aryabhata** est un mathématicien et célèbre astronome indien né en **476**. Contrairement à Aristote, Ptolémée, et aux autres philosophes de l'époque précédente, pensants que la Terre serait immobile ; Aryabhata quant à lui, affirma la rotation de cette dernière. Dans son traité de mathématiques, il explicite des algorithmes permettant de trouver la racine carrée et cubique, découvert par Héron auparavant. Il serait aussi le premier à avoir utilisé la demi-corde, ancêtre du sinus, pour le calcul d'angles dans un triangle. Pour solutionner plus rapidement les calculs trigonométriques, il réalisa une table précise de 0 à 90 degrés.

Après la chute de l'Empire romain, c'est celui des Perses qui s'étend et provoque un **âge d'or de l'islam** qui dura du 8^{ème} au 13^{ème} siècle. Je vais donc introduire plusieurs savants de cette époque qui ont eu accès aux textes antiques grâce à la conquête d'Alexandrie ou d'Antioche vers 650. Ils ont transmis et développé les sciences à la manière de leurs prédécesseurs, jusqu'à ce que les Espagnols reprennent peu à peu leurs territoires lors de la Reconquista.

L'*algèbre* est l'ensemble de règles qui permet d'écrire des formules et équations mathématiques afin de structurer un problème en un système calculable. Un algorithme quant à lui est une suite finie et claire de tâches à réaliser pour résoudre un problème. Le plus ancien algorithme connu est la recette de cuisine. Les premières recettes « publiées » datent de l'époque babylonienne : trois tablettes, conservées à l'université Yale et datant d'environ 1600 ans avant J.-C. Elles comportent de manière plus ou moins précise une série de recettes de cuisine. Similairement, les mathématiques de l'époque ont détaillé des procédures permettant de résoudre des problèmes étape par étape. Il a fallu cependant attendre le 9^{ème} siècle, vers l'an **800**, pour que **Al-Khwarizmi** définisse clairement l'**algèbre**, qui signifie réparer une fracture, et sert par définition à la résolution d'un problème. Le livre qu'il a écrit, *Abrégé du calcul par la restauration et la comparaison*, avait pour vocation d'apporter des solutions à l'héritage, l'arpentage, et les échanges commerciaux. De surcroît, il a démocratisé le concept d'équation, égalité entre deux expressions mathématiques, ainsi que sa manière de les rédiger en langage mathématique, permettant la traduction d'un problème en une formule courte, avec la solution représentée en tant que variable inconnue. La résolution de l'équation passe par une suite d'opérations, résultant idéalement en un ensemble ou système d'équations de la forme `variable_inconnue = [résultat_numéraire]`. Le mot **Algorithme** est d'ailleurs né d'une longue déformation du nom « Al-Khwarizmi » par les traducteurs latin en « Algoritmi » au 12^{ème} siècle. Hormis l'algèbre et les algorithmes, il nous a transmis la base même de nos mathématiques. Le **système de numération indo-arabe** est la notation des chiffres encore utilisée aujourd'hui. Elle provient du monde arabe oriental, qui à partir de la fin du 7^{ème} siècle, entreprends un mécénat scientifique s'intéressant principalement aux sciences indiennes. Des bibliothèques se forment alors, et bon nombre de textes anciens sont alors traduits.

Trois frères appelés **Banou Moussa** sont des savants arabes ayant étudié les mathématiques, l'architecture, la mécanique et l'astronomie. Deux de leurs ouvrages nous sont parvenus, le *Livre sur la détermination des surfaces des figures planes et sphériques*, ainsi que le *Livre des mécanismes ingénieux*, qui ont été publiés à la **Maison de la sagesse** ouverte en **832** à Bagdad. Cet établissement est à l'origine du mécénat décrit précédemment. Pendant le 9^{ème} siècle, plein de savants y travailleront, dont **Al-Kindi**. Ayant côtoyé Al-Khwarizmi et les frères Banou Moussa lors de la traduction de multiples œuvres grecques en arabe, il sera imprégné par la pensée aristotélicienne. Son œuvre la plus notable est son *Manuscrit sur le déchiffrement des messages codés*, première œuvre connue à traiter le sujet de la **cryptanalyse**. Il y indiquera notamment l'analyse de fréquence d'un caractère, permettant de facilement retrouver la clé d'un chiffrement par décalage comme le Code César. Enfin vient **Al-Battān**, un astronome et mathématicien arabe souvent considéré comme le « Ptolémée arabe ». Il a en effet repris les travaux de Ptolémée pour les compléter afin de constituer des tables de calculs pour la position du Soleil et de la Lune. Ses tables ont longtemps été utilisées et elles ont influencé l'astronomie européenne, y compris les travaux de Kepler, Copernic et Galilée. Il a entre autres découvert le mouvement de l'apogée du Soleil, recensé plus de 500 étoiles, calculé les équinoxes et l'inclinaison de l'axe terrestre, puis démontré que la distance entre le Soleil et la Terre varie lors d'une révolution, et enfin affiné les calculs de Thalès en montrant qu'une année est constituée de 365 jours, 5 heures, 48 minutes, et 24 secondes.

Alors que cet âge d'or est proche de son apogée, ses savoirs se sont petit à petit occidentalisés. Sylvestre II, né en **950** et mort en 1003, aussi connu sous le nom de Gerbert d'Aurillac, aurait demandé à l'astronome Lupitus de Barcelone, un traité sur les calculateurs analogiques permettant de mesurer la hauteur des étoiles, aussi appelés **astrolabes**. Les historiens ne savent pas si elle lui est parvenue, mais il a introduit des concepts de la science arabe en Occident. Lors d'un séjour de 3 ans en Catalogne, il aurait en effet consulté des manuscrits traduits de l'arabe, dans lesquels il aurait pu avoir pris connaissance des chiffres indo-arabes, ainsi que l'écriture décimale positionnelle. Il a inventé un abaque dont une version pourrait avoir été réalisée avec cette notation des chiffres. Après lui, Hermann Contract, né en **1013** et mort en 1054, aboutira ses travaux sur l'astrolabe et en concevra même un. Malgré une paraplégie spastique familiale, handicap de naissance affectant sa parole et sa lecture tout en l'empêchant de marcher, il a permis, à l'instar de Vitruve, la transmission de bien des savoirs concernant autant l'Histoire, la musique, les mathématiques, l'astronomie et la poésie.

Forgeron de formation, né à Tolède en **1027** et mort en 1087 à Cordoue, **Al-Zarqālī** s'intéressa à l'astronomie et grava des astrolabes. Il effectua des observations du ciel et réalisa des tables sur le mouvement des planètes nommées **Tables Tolédanes**. La précision de ses tables permettait la prédiction d'éclipses. Il aurait également repris les travaux de Ptolémée et Al-Khwarizmi pour en corriger leurs résultats. Deux siècles après sa mort, Alphonse X de Castille ordonna la traduction de toutes ses œuvres littéraires et commandita la réalisation de tables

alphonsines permettant le calcul de la position des astres tels que le Soleil, la Lune et les planètes. Ces travaux seront ultérieurement repris par [Copernic](#).

De 722 à 1100, l'Espagne reprend jusqu'à la moitié de ses territoires occupés jusque là par les peuples arabes. Probablement né en **1070**, **Abraham bar Hiyya Hanassi** y a vécu jusqu'au début du 12^{ème} siècle et s'y éteindra vers l'an 1140. Rabbín, mathématicien, astronome et philosophe, il est parfois considéré comme le véritable pionnier des sciences mathématiques en Europe. Il a traduit l'*Algèbre* d'Al-Khawarizmi en latin, avec l'aide de Platon de Tivoli, dans une œuvre nommée *Liber embadorum*, qui servira à Fibonacci.

Averroès était un médecin philosophe né en **1126**. Très influencé par Aristote, il sera un fervent défenseur de la pensée logique et de Galien. Ce dernier déclarait lui-même s'appuyer à la fois sur la raison et l'expérience qu'il surnommait « ses deux jambes ». À la manière de ce dernier, il rédigea des traités, dont un de pharmacologie, mais surtout quantité de commentaires de textes antiques pour lesquels il sera considéré comme un compilateur commentateur. Il succède à Avicenne, un autre philosophe médecin perse qui rédigea trois encyclopédies, une de philosophie, de sciences, et de médecine. Ses travaux seront repris en tant que livres de connaissances, et ceux d'Averroès comme des livres d'exercices, car ses commentaires étaient remplis de questions pouvant servir d'entraînement à des élèves.

Comme bien d'autres, les historiens ne savent pas grand-chose de sa vie, mais **Al-Jazari** fut actif vers **1206**, où il aurait écrit, après 25 ans d'études, le « *Livre de la connaissance des procédés mécaniques* », à la demande de Nasir Al-Din Mahmoud, le prince artukide de l'époque. Sûrement inspiré du *Livre des mécanismes ingénieux* des frères Banou Moussa, eux même inspirés des travaux des mécaniciens de l'Antiquité ; il transmettra à travers ce qui sera le traité de mécanique le plus important du monde arabe, des informations sur les pompes et machines hydrauliques, les automates, la manivelle, et bien d'autres ; illustré notamment avec un automate verseur de vin, qui n'est pas sans rappeler la servante de [Philon de Byzance](#).

Ayant vécu de **1170** à **1250**, **Leonardo Fibonacci** serait le « chaînon manquant » qui aurait importé la notation indo-arabe aux mathématiques occidentales, à une période où les chiffres romains prédominent. Il a en effet été éduqué à Béjaïa en actuelle Algérie, et aurait ramené entre autres cette fameuse notation à Pise, où son père était marchand et notaire public des douanes. Il est réputé pour sa fameuse suite, liée au nombre d'or, proportion qu'il n'a pas évoquée, mais commune à de vastes formes, y compris des structures produites par la nature.

Opposé à Averroès, le philosophe espagnol **Ramon Llull**, né en **1234**, est parfois considéré comme un illuminé. Il dira avoir rencontré Dieu, qui lui aurait donné un système lui permettant de mémoriser l'ensemble de sa connaissance. Ce système fait de disques concentriques comportant des symboles est une création épistémologique relationnelle, basé sur des paires et des triples. C'est-à-dire qu'elle représente le savoir ou du moins des idées, grâce à des combinaisons. [Trois siècles après, Leibniz](#) adoptera cette idée qu'il repensera comme un « alphabet de la pensée logique ». Cet alphabet sera repensé en binaire par [George Boole au](#)

19^{ème} siècle. En tout cas le visionnaire Ramon Llull découvrit une multitude d'idées reprises dans l'informatique, comme la théorie des graphes, l'alphabet et ses logiques combinatoires. De plus, en 2001, la découverte des manuscrits perdus *Ars notandi*, *Ars eleccionis* et *Alia ars eleccionis*, sera reconnue comme l'origine du paradoxe de Condorcet qui ne sera popularisé qu'à la fin de l'époque moderne par le marquis du même nom.

D'un point de vue social et économique, l'humanité connut une forte croissance démographique à cette période, avec un doublement de la population passant de 35 à 80 millions d'individus entre l'an 1000 et 1350, malgré la grande famine de 1316 et la peste noire de 1347. Ce développement fut en grande partie dû à un climat plus favorable et à l'agrandissement des surfaces cultivées, qui a été accompagné par le développement des techniques agricoles comme la rotation des cultures ou l'apparition de la charrue. L'introduction du rouet chinois en Europe permit la réalisation de textiles de meilleure qualité. Les moulins à vents et à eau proliférèrent, permettant une automatisation et la réduction de l'utilisation de forces manuelles au profit des forces hydrauliques, thermiques, éoliennes et animales. Les techniques navales se sont développées avec les coques bordées en clin à la manière des tuiles d'un toit, puis à franc bord où les planches sont jointes comme du parquet ; mais aussi avec la popularisation des voiles latines triangulaires et des gouvernails d'étambot articulés. La réalisation de cathédrales et de châteaux a accéléré le développement des techniques de construction, des armures, et des armes, notamment de siège. Les hauts fourneaux permettant l'obtention de fonte et d'alliages de qualité apparaissent en Suède vers la fin du 13^{ème} siècle, peu avant les armes à feu.

Concernant les outils informationnels, cette période a notamment eu une invention dans la continuité des claviers inventés par Ctésibios. En grec, *týpos* signifie frappe, coup, ou pression, et peut même indiquer la marque qui en résulte. La **typographie** est donc l'art d'écrire en appliquant des caractères mobiles, tampons munis d'un symbole appelé glyphe. Un ensemble de glyphes représentant un alphabet complet forme une fonte ou police de caractères. Avec ces derniers il est donc possible de créer des mots, des phrases, et de les imprimer. Cette technique a été détaillée vers **1440** par **Gutenberg**, qui n'a pas inventé l'imprimerie, mais l'a popularisée avec l'ensemble des techniques mécaniques liées. En effet, les caractères mobiles existaient depuis leur invention en 1040 par un inventeur chinois nommé Bi Sheng ; et la xylographie permettait déjà l'impression à l'aide de gravures sur bois, malgré son usure après plusieurs utilisations. La popularisation de l'imprimerie a permis l'automatisation d'une transmission écrite qui était alors manuelle, réduisant les erreurs et les coûts tout en améliorant la vitesse de production et de diffusion des informations. La lecture qui était alors un privilège réservé aux plus riches devient alors accessible, la transmission orale ne prévaut plus et chacun peut avoir un libre examen des œuvres littéraires.

La fin de cette époque a aussi été florissante pour les Italiens, le polymathe **Leon Battista Alberti** est considéré par certains comme le père de la cryptographie occidentale, il a en effet inventé le cadran chiffant, un système de chiffrement par substitution, et rédigé un texte prouvant que l'analyse de fréquence d'apparition des lettres dans les textes en permet le déchiffrement. Il finira donc logiquement par inventer le surchiffrement codique, l'innovation cryptographique la plus significative depuis la période de César. Cette méthode consiste à utiliser plusieurs alphabets désordonnés en alternant entre eux lors du chiffrement. Il aurait imaginé le premier anémomètre permettant d'évaluer précisément la force du vent, et finira par décéder en 1472 à Rome. À la fin du 15^{ème} siècle, un autre habitant de Rome nommé **Luca Pacioli**, vulgarise les mathématiques et reprends le concept de *nombre d'or* dans un livre qui participera au mythe qui lui est lié. Cette œuvre appelée « *De divina proportion* » sera illustrée par le peintre polymathe **Leonard de Vinci**, qui de son vivant réalisera plusieurs machines de théâtre comme de guerre, transmettant par la même occasion des prototypes de machines volantes à ses successeurs. L'avancée la plus significative que Luca Pacioli ait apportée à l'humanité est cependant la comptabilité par partie double, toujours utilisée aujourd'hui en entreprise. Du reste, il a traduit *Éléments* d'Euclide en Latin, et publié un résumé d'arithmétique de proportion et de géométrie, à Venise en 1492.

Époque moderne

Le début en étant marqué par la découverte occidentale d'un nouveau continent en **1492**, cette période sera notamment celle du développement des empires coloniaux et des grandes découvertes maritimes, permit par les développements technologiques du Moyen-Âge, notamment concernant la navigation, la cartographie, l'imprimerie et les techniques agricoles. L'import de la pomme de terre américaine permettra d'atténuer le problème des famines. C'est à ce moment que s'amorce un changement sociétal, avec le très lent déclin des monarchies, du clergé et de la noblesse, au profit de la bourgeoisie qui développa le commerce, notamment dans les villes et leurs abords, causant une urbanisation progressive. L'Église perd petit à petit de son pouvoir, et ses biens entrent progressivement dans le domaine public grâce à la sécularisation. La pensée scientifique logique va de même connaître un tournant, des informations connues depuis l'Antiquité vont alors s'imposer difficilement, mais finir par être démocratisées. Aristarque de Samos vécu vers -280. C'est la première personne connue à envisager que le Soleil soit au centre de notre univers et que la Terre tourne autour. Mais c'est Nicolas **Copernic** qui vers **1513** proposa un modèle héliocentrique viable. Par la suite, Johannes **Kepler** a étudié et complété ce système en ajoutant que les trajectoires des planètes autour du soleil sont en réalité elliptiques. À cet effet, il rédigea trois lois éponymes régissant les orbites, les aires, et les trajectoires. **L'héliocentrisme** a ensuite été confirmé par **Galilée** qui a perfectionné des lunettes astronomiques grâce auxquelles il réalisa les observations nécessaires, et à partir desquelles il rédigea les premiers principes mécaniques permettant de justifier ce système. Enfin, dans son ouvrage *Philosophie naturelle et principes*

*mathématiques*⁸ créé en 1680 et paru en 1687, **Isaac Newton** a abouti les travaux de Copernic, Kepler, et Galilée pour établir les **lois de la gravitation universelle**. C'est ainsi le fondateur de la mécanique classique et des trois lois portant son nom. D'autre part, il prouva la décomposition de la lumière ainsi que son spectre lumineux à l'aide d'un prisme⁹. Après cela il faudra attendre **1757** pour que trois calculateurs français, Alexis Clairaut, Jérôme Lalande, et Nicole Lepaute aboutissent le travail d'un polymathe anglais qui répondait au nom d'Edmond **Halley**. Ce dernier expliqua dans un livre que ce que les astronomes croyaient être des **comètes** distinctes n'en est en réalité qu'une seule, ayant une périodicité de 76 ans pour effectuer une révolution autour du soleil. Ce travail conjoint a permis l'abandon définitif de la théorie des tourbillons de Descartes au profit de la mécanique newtonienne. Copernic est ainsi à l'origine du mouvement philosophique des Lumières, dont feront partie la plupart des philosophes lui succédant comme Descartes, Spinoza, Newton, et cetera... Ce mouvement a donné lieu au siècle des **Lumières**, qui a vu émerger de grandes innovations technologiques, avec un détachement de l'Église et de son enseignement théologique inspiré de la philosophie d'Aristote, c'est-à-dire la scolastique, au profit de la raison et des sciences.

Révolutions informationnelles

Giambattista della Porta était un écrivain polymathe fasciné par l'ésotérisme derrière les miracles et mystères de la nature. Il œuvra à les opposer au divinatoire, et ainsi convertir des croyances en savoirs scientifiques. Comme plein de philosophes et esprits polymathes, sa pensée était dirigée par les principes préscientifiques de la *théorie des analogies et de la correspondance*. Il a étudié l'optique et le magnétisme, à travers les lentilles et l'attraction du fer sur un aimant, ainsi que la propriété de nombreux métaux. Étant cryptographe, il rédigea un ouvrage¹⁰ détaillé résumant les connaissances de **cryptanalyses** connues à l'époque. Publié en **1563**, il y traita du chiffrement et déchiffrement de messages, avec quelques ajouts de sa part, comme le système littéral à double clé qui sera longtemps utilisé, et dont il est potentiellement l'inventeur. De plus, il participa à l'aboutissement des travaux de [Leon Battista Alberti](#) pour en faire un système cryptographique complet. De surcroît, il s'intéressa à la psychologie et notamment la mémoire, domaine qu'il étudia jusqu'à concevoir des astuces mnémoniques utilisées par les acteurs de théâtre pour mémoriser leurs textes. Blaise de **Vigenère**, né le 5 avril 1523 à Saint-Pourçain-sur-Sioule, était secrétaire d'ambassade. Il n'a pas inventé le **code** portant son nom, qui a été introduit par Giovan Battista Bellaso. Cependant, Vigenère le popularisa dans son « *traité des chiffres* » publié en **1586**, décrivant précisément son chiffrement avec une table qui sera utilisée pour déchiffrer les messages, jusqu'à ce que le code ne soit rendu obsolète par le major prussien Friedrich Kasiski qui l'a cassé dans une publication de 1863. Au lieu d'utiliser un simple décalage comme le ferait un [Code de César](#), une clé-mot est utilisée pour chiffrer le message en associant un chiffre à chaque caractère du mot, indiquant

le décalage à réaliser. Cette suite de décalage obtenue à partir de la clé-mot cryptographique est ainsi répétée pour chiffrer l'intégralité du texte.

La base de l'informatique repose aussi sur un chiffrement de la donnée. Le **binaire** est pratiqué depuis l'an -750 avec les hexagrammes chinois. Vers l'an -200, le poète et mathématicien indien **Pingala** rédigea *Chandahsastra*, étymologiquement « Code de la prosodie ». Il y étudiera mathématiquement la construction des phrases des poèmes sanskrits, par combinaisons de syllabes longues ou brèves. Formulant ainsi le texte connu le plus ancien traitant des nombres binaires, et de la métrique ou versification poétique. Ce n'est pourtant qu'au **17^{ème} siècle** que **Juan Lobkowitz** sera considéré comme le fondateur de la mathématique binaire, dont le concept et ses opérations tels qu'ils sont utilisés aujourd'hui, n'ont été formalisés qu'en **1690** par **Leibniz Wilhelm Gottfried**. Il a popularisé ce système en démontrant sa facilité d'écriture et d'usage, notamment pour la division qui était à ce moment compliquée à automatiser. Il a aussi projeté son utilisation future en émettant l'idée que des machines plus élaborées puissent en tirer pleinement profit. C'était un grand polymathe allemand qui traita pléthore de sujets comme la Géologie, Biologie, Médecine, Physique, Philosophie, Logique et les Mathématiques. Son invention la plus notable liée à l'informatique est le **cylindre cannelé**, élément mécanique strié avec des barres longitudinales de plus en plus grandes. Plus il est enfoncé, plus il y a de barres qui s'enclencheront dans une roue juxtaposée, servant ainsi de mémoire pour stocker un nombre de barres qui servira de **multiplicateur**. Enfin, il a inventé une langue logique universelle et formelle, voire mécanisable, la *characteristica universalis*.

Le terme « computer » a été écrit pour la première fois par Richard Brathwaite dans le livre *The Yong Mans Gleanings* en **1613**, sauf que le terme ne faisait pas référence à une machine, vous devriez pouvoir à ce stade de la lecture, constater que les ordinateurs d'antan étaient des gens, qui calculaient et rédigeaient des tables de calcul (logarithmiques, trigonométriques, et cetera...), et ce afin d'avoir la réponse à un calcul de manière directe c'est-à-dire avec une complexité constante en O^1 . Les premières machines à calculer sont apparues au courant du 17^{ème} siècle, mais avant cela apparaissent de nouveaux outils de calcul non mécanisés. C'est ainsi en **1617** que John Napier, un mathématicien écossais qui a donné son nom au *logarithme népérien*, invente un abaque du même nom nommé **Bâtons de Napier**. Il facilite le calcul des produits, quotients, puissances et racines. Cet outil permet, à l'aide d'un **tableau**, ayant pour lignes les chiffres de 1 à 9, et pour colonnes les chiffres du nombre sur lequel je désire faire une opération, d'obtenir une liste de cases donnant le résultat de l'opération.

Son affichage se matérialise par des cases séparant les dizaines, situées en haut à gauche, des unités, placées en bas à droite et toutes deux séparées par un trait. L'utilisateur obtient le résultat de l'opération avant application des retenues. Il suffit alors d'additionner les nombres situés dans les mêmes diagonales pour finir le calcul. Sur la figure ci-dessous, je multiplie 1 234 567 890 par 6 et obtient ainsi 7 407 387 340.

		Chiffres											
Multiplicateurs		1	2	3	4	5	6	7	8	9	0	Résultats	
	1	0	2	3	4	5	6	7	8	9	0		
	2	0	4	6	8	1	2	3	4	5	6		
	3	0	6	9	1	2	3	4	5	6	7		
	4	0	8	1	2	3	4	5	6	7	8		
	5	0	1	2	3	4	5	6	7	8	9		
	6	0	2	3	4	5	6	7	8	9	0		
	7	0	3	4	5	6	7	8	9	0	1		
	8	0	4	5	6	7	8	9	0	1	2		
	9	0	5	6	7	8	9	0	1	2	3		

X	1	2	3	4	5	6	7	8	9	0
6	6	12	18	24	30	36	42	48	54	0
7	7	14	21	28	35	42	49	56	63	0
10	10	20	30	40	50	60	70	80	90	0
7407387340										

6 X 1234567890 = 7407387340

F3 : Bâtons de Napier

Pour clôturer la présentation d'abaques, c'est en **1621** que **William Oughtred**, qui est à l'origine de la notation de π et « x » pour la multiplication, se base sur les travaux de Napier et invente une **règle coulissante** destinée à calculer des multiplications, divisions ainsi que des exponentielles, racines, puissances et calculs trigonométriques, laissant l'addition et la soustraction à de plus simples abaques. Aujourd'hui obsolètes, elles ont pendant longtemps été, à la manière des tables de calcul, une solution suffisamment précise, abordable et facile à créer.

C'est au début du 17^{ème} siècle que **Wilhelm Schickard** inventa une **horloge à calculer** avec l'aide des travaux de Napier à qui il dédia une éphéméride. Malheureusement, lors de sa conception un incendie vint détruire ses avancées, et une reproduction fonctionnelle montra qu'il manquait certains moyens techniques pour finaliser son œuvre et la rendre opérationnelle. Son invention ne sera donc pas retenue par les historiens comme la première machine à calculer, mais plus comme un prototype avancé. En **1650**, le clermontois polymathe **Blaise Pascal**, invente la **Pascaline** trois ans après son traité de géométrie projective, alors qu'il n'a que 19 ans. Cet outil est aujourd'hui considéré comme **la première machine à calculer**. Elle a été réalisée dans la volonté de soulager le travail de son père, nommé premier président à la Cour des aides de Normandie à Rouen. Cette machine a permis de réaliser additions, soustractions, et multiplication, ainsi que divisions, par répétitions. C'était la seule machine à calculer fonctionnelle au 18^{ème} siècle, elle marque le début d'une période de développement de machines à calculer de plus en plus sophistiquées. C'est la première fois que l'humanité remarque un impact de l'automatisation notable sur l'emploi et la société, associée à une technophobie, partiellement justifiée par la création de machines de guerre dans le passé.

Les **cartes perforées** sont des morceaux de papier ou de carton, souples ou rigides. Elles comportent des grilles dont la présence ou l'absence de trou correspond à une information binaire. Son utilisation la plus ancienne est avérée dès **1502** avec le premier **orgue de Barbarie**, ancêtre de la boîte à musique. À partir du 16^{ème} siècle, des automates sont réalisés avec cette technologie. En **1725**, le Lyonnais Basile Bouchon met au point le premier système de programmation d'un **métier à tisser** semi-automatique à l'aide d'un fragile ruban de papier perforé permettant de le programmer. Trois ans après, en 1728, son assistant nommé Jean-Baptiste Falcon, a l'idée de remplacer le ruban par une série de solides cartes perforées en

carton reliées entre elles, améliorant la robustesse de cette machine. Vers 1750, le Grenoblois inventeur d'automates Jacques Vaucanson, automatise la machine à l'aide d'un système hydraulique, tout en remplaçant les cartes perforées par un cylindre métallique à pointes.

De 1751 à 1772, Diderot et Alembert ont rédigé l'*Encyclopédie* ou le *Dictionnaire raisonné des sciences des arts et des métiers*. J'y ai retrouvé l'une des premières personnalités connues à inventer une nouvelle signification au mot **système**. Mort en 1707, Sébastien Le Prestre, plus connu sous le nom de **Vauban**, dont il était le marquis, a établi le système dans le sens de l'art militaire, le décrivant comme l'arrangement d'une armée ou la disposition de toutes les parties d'une fortification par un général ou un ingénieur¹¹. De manière générale, l'étude des systèmes est celle des ensembles complexes d'interactions entre des sous-groupes. Dans cette même Encyclopédie se trouvent les définitions du système en anatomie, poésie, musique et en finance. Après Vauban, Étienne Bonnot de **Condillac** a défini les prémices de l'approche systémique dans le cadre de la science politique à l'aide de son *Traité des Systèmes* en 1749, par la suite repris par Vilfredo **Pareto** qui l'appliqua à l'économie politique au début du 20^{ème} siècle. Le 18^{ème} siècle est donc la source de la théorie générale des systèmes et de la systémique, qui étudient des phénomènes comme faisant partie d'un tout desquels ils émergent, c'est à dire qu'ils abordent les situations et acteurs comme étant imbriqués et interconnectés dans un réseau ou une organisation complexe.

Pour faire la transition avec l'époque suivante, je vais brièvement parler du journalisme. Dans l'Antiquité, les gens le pratiquaient à l'oral, des crieurs, messagers ou troubadours, annonçaient les faits importants sur la place publique. Le journalisme écrit trouverait potentiellement ses racines dans l'Empire romain avec les *Acta Diurna*, traductible en l'ordre du jour. La République de Venise ou La Dynastie Han ont également publié des bulletins d'information. Cependant le journal comme évoqué aujourd'hui, imprimé en masse, périodique et populaire, naîtra en 1622 avec le *Weekly News* de Nathaniel Butter. Il sera rapidement copié et récupéré en France, où le journalisme est né grâce à Théophraste Renaudot et sa Gazette de 1631. L'histoire du journalisme est très liée au courant philosophique libéral introduit par Bernard Mandeville, puis John Locke, et Adam Smith. La liberté de la presse s'imposera contre la censure royale et sera même inscrite dans les droits de l'Homme et du citoyen de 1789. Les journaux et publications se développeront alors énormément jusqu'à la période suivante, nécessitant des « grossistes » de l'information. L'*Agence des feuilles politiques, correspondance générale* a été créée en 1835 par Charles-Louis Havas. C'est la première agence de presse qui deviendra la fameuse Agence France Presse toujours existante et faisant référence, gage de qualité et de véracité de l'information. La liberté d'expression et de la presse sera réellement actée avec les lois de 1881. Au-delà de ses droits, le journalisme a aussi le devoir de fournir une information fiable en vérifiant les faits, et de protéger ses sources afin de permettre aux lanceurs d'alerte de ne pas être persécutés.

Époque contemporaine

C'est la période actuelle, le début en est défini par la révolution industrielle qui a commencé en **1760** au Royaume-Uni, le besoin de calcul, de gestion et d'automatisation est alors grandissant. La France abolie la monarchie en 1792, les États deviennent pour la plupart des républiques, l'ancien régime et les empires coloniaux prennent fin, l'esclavage est aboli, et les deux guerres mondiales prendront lieu par la suite. Depuis la fin de l'époque moderne et pendant tout le 19^{ème} siècle s'est développé l'éducation nouvelle et la **pédagogie**, avec ses variantes laïques, gratuites, actives et mutuelles. Je ne pourrais pas détailler ses principaux acteurs que sont Charles Démia, Jean Jacques Rousseau, Johann Heinrich Pestalozzi, Grégoire Girard, Andrew Bell, Joseph Lancaster, Francisco Ferrer, Adolphe Ferrière, Maria Montessori, Nicolas de Condorcet, Louis-Joseph Charlier et bien évidemment Jules Ferry. Je tiens juste à les mentionner pour que vous puissiez vous y (ré)intéresser. Ceux qui m'ont le plus marqué sont Célestin Freinet avec la correspondance et son journal, ainsi que John Dewey, auteur de *The School and Society*¹², qui a été imprégné par la théorie de l'évolution de Charles Darwin, et le pragmatisme du logicien Charles Sanders Peirce dont je parlerai dans ce chapitre.

De la mécanique à l'électronique

La dactylographie est, comme son étymologie grecque l'indique, écrire avec les doigts. En 1714, Henry Mill breveta la première machine à écrire, mais les historiens ne savent pas si elle a été construite et utilisée. Par la suite, William Austin Burt crée The Typographer en 1829, mais je n'ai trouvé que peu d'information là-dessus. Il faudra attendre **1837**, année à laquelle un italien nommé Guiseppe Rivezza réalise le **clavecin scribe**. Cette machine inspirée des pianos sera la plus complète jusqu'à l'apparition de la Remington. Entre-temps Jean Claude Pingeron et François Pierre Foucault, ami de Louis Braille, vont respectivement en 1780 et 1839, inventer deux raphigraphes. Ce sont des machines permettant aux aveugles de taper des textes en braille avec leurs doigts. Cette technologie a facilité le dialogue entre les voyants et les non-voyants, permettant notamment aux aveugles de communiquer plus facilement avec des membres de leurs familles. L'entreprise Remington Arms créée en 1816, fabriquait initialement, comme son nom l'indique, des armes à feu, puis du matériel agricole et des machines à coudre. En **1874** cette entreprise créera la « **type-writer** » **Remington**, elle nous a apporté les claviers QWERTYUIOP, et la suite alphabétique présente sur la majorité des claviers, FGHJKL. Une vingtaine d'années plus tard, la pratique sera assez démocratisée pour que Georges Buisson alors sténographe à l'Assemblée nationale, organise et réalise des concours de vitesse dactylographique. Enfin, *Les aventures de Tom Sawyer* de Mark Twain, sera la première œuvre littéraire à être écrite intégralement sur machine à écrire lors de sa publication en 1872.

Depuis **1766**, l'Almanach nautique est édité chaque année, cette bible du marin est le premier projet de **table permanent**. Avant d'être automatisée, elle était calculée par deux personnes différentes et validée par une dernière qui comparait les résultats. Lorsque deux des auteurs principaux, les astronomes Malachy Hitchins et Nevil Maskelyne meurent respectivement en 1809 et 1811, l'ouvrage sombre pendant 20 ans, croulant sous les erreurs. La Pascaline a longtemps été la seule machine à calculer fonctionnelle, plein de prototypes seront construits pendant le 18^{ème} siècle, mais ce n'est qu'au début du 19^{ème} que s'opère la production industrielle de calculateurs mécaniques, qui viendra répondre aux problèmes d'éditions de tables et en faciliter la création comme la vérification. L'**Arithmomètre** développé par Thomas de Colmar en **1820**, sera la première machine à addition commercialisée. Elle n'a cependant jamais été produite en grande quantité, car elle était réalisée à la main à raison d'un ou deux exemplaires par mois. De plus, le procédé de calcul était très lent, la plupart des utilisateurs n'y voyaient pas de gain de temps lors de petits calculs, rendant son utilisation quasiment exclusive aux assurances et ingénieurs, qui ont recours à des calculs mettant en œuvre de grands nombres, supérieurs au million. Malgré son bas coût de 150\$, la demande pour ce produit resta donc faible. Une vingtaine d'années plus tard apparaît l'**Arithmaurel**, créé en **1842** par Timoleon Maurel, avec l'Arithmomètre et le cylindre de Leibniz pour inspiration. Cette machine à calculer était utilisable en renseignant simplement les valeurs (opérandes) et les opérateurs. C'est une grande avancée en termes d'expérience utilisateur. Malheureusement sa conception à grande échelle n'a pas été possible à cause des limitations techniques de l'époque. La division quant à elle restait fastidieuse, car elle demandait à l'utilisateur d'effectuer un ensemble de soustraction avec beaucoup d'attention. Des calculateurs analogiques très spécifiques feront leur apparition, comme des planétaires, ou la **machine à prévoir la marée** de Lord Kelvin créée en **1872**. Elle a permis de réaliser des tables de marées pour divers ports, évitant ainsi aux bateaux de s'échouer sur les rochers ou les plages. Pour reprendre sur les calculateurs mécaniques, Dorr E. Felt dévoile le **comptomètre** en **1887** aux États-Unis. Il l'a réalisé en reprenant l'idée du clavier à touches de l'arithmomètre. En **1889**, le mécanisme des **calculatrices à crosse** est breveté par le Français Louis Troncet, sous le nom d'**Arithmographe**. Pour finir, la plupart de ses machines auront leur version américaine réalisée à partir de la fin du 19^{ème} siècle. En effet **William S. Burroughs** fonda l'American Arithmometer Company en 1886, entreprise qui finira par s'appeler éponymement Burroughs. Cette entreprise aura un grand succès avec la réalisation de **caisses enregistreuses**. Ces machines à calculer primitives ont rapidement changé avec le développement de tout un champ de recherche qui naît dans l'époque contemporaine.

Depuis les penseurs grecs de l'Antiquité, l'humanité a connaissance de l'**électricité** statique résultant notamment du frottement de l'ambre. En effet *elektron* signifie ambre jaune en grec ancien, et le mot magnétisme viendra de la magnésie, pierre servant d'aimant naturel depuis la nuit des temps. En **1745**, juste avant l'époque contemporaine, **Pieter van Musschenbroek**, professeur de physique dans la ville de Leyde aux Pays-Bas, découvre qu'il est possible de stocker des charges électriques dans deux électrodes opposées par un isolant, créant ainsi un

ancêtre du **condensateur** nommé **Bouteille de Leyde**. Ce composant électrique est capable de renvoyer des décharges spectaculaires qui ont été utilisées pour divertir les gens dans les foires, ou des centaines de courtisans de Louis XV et lui-même à Versailles. Des appareils de mesure des charges électriques voient alors le jour, comme l'électroscope puis l'électromètre. **Charles Coulomb** étudie à son tour les phénomènes électrostatiques. En 1785, il découvre la **formule de l'intensité de la force électrique** entre deux particules chargées, représentée par le produit de leurs charges et le segment entre elles, plus sa distance est courte, plus la force sera grande. La découverte de la **pile électronique** de **Volta** en **1800** a quant à elle instantanément reçu une renommée internationale. L'intérêt pour ce domaine de recherche se développe alors à une vitesse folle. En 1820, Hans Christian Orsted montre qu'un courant électrique peut dérégler une boussole, dont le principe est connu depuis le 1^{er} siècle et popularisé vers le 10^{ème}. Onze ans plus tard, en 1831, **Michael Faraday** énonce une loi éponyme permettant de relier le courant électrique aux champs magnétiques. Cette loi fut complétée par les travaux d'**André-Marie Ampère** qui proposa un théorème reliant le champ magnétique aux courants électriques. Enfin, en 1873 **James Clerk Maxwell** unifie définitivement les champs électriques et magnétiques dans la **théorie de l'électromagnétisme**. Entre-temps apparaissent des applications concrètes comme la roue de Barlow, premier moteur électrique créé en 1822, le premier électroaimant en 1825, l'induction en 1831, et l'année suivante, le premier **alternateur** nommé machine de Pixii. Ces technologies seront améliorées à force d'itérations, ouvrant nombre de possibilités comme la suppléance des machines à vapeur par des machines dynamoélectriques.

Thomas Edison récupère le brevet pour la lampe à filament en 1879, qu'il n'a donc pas inventé, mais démocratisé et amélioré, devenant malgré ceci le pionnier dans les lampes électriques. Il était donc essentiellement un homme d'affaires, considéré comme l'inventeur le plus prolifique. De reste, il est à l'origine des premiers bureaux de recherches industrielles, ainsi que des films vidéos. **Nikola Tesla** a travaillé avec lui, mais il finit pourtant son parcours en indépendant. Il a inventé le moteur asynchrone comme l'**alimentation polyphasée**¹³, et a travaillé sur le **réseau de distribution d'électricité** basé sur cette même technologie alternative. Après quoi il s'est fortement intéressé à la très haute fréquence et à la **technologie sans fil**, espérant un jour pouvoir transmettre de l'électricité à travers l'océan Atlantique. Il s'est brièvement intéressé aux **rayons X**, faisant de lui un des pionniers de l'**imagerie radio**, peut être même le premier. Mais après avoir appris que **Wilhelm Röntgen** avait découvert les rayons X quelques mois après lui, et que les dangers de ce type de rayonnement se fassent sentir chez lui et son assistant, il décide alors de s'orienter dans les automates radio-contrôlés, qui feront de lui le premier concepteur de robot télécommandé. **Lucien Gaulard** a par la suite amélioré les travaux de Tesla sur l'alimentation polyphasée en inventant le **transformateur**, capable d'augmenter la tension d'un alternateur pour faciliter le transport de l'énergie dans des lignes à haute tension. Enfin, **Albert Einstein** naquit l'année où Edison déposa son brevet pour la lampe. Il popularisa son concept de la **relativité générale** et tenta d'unifier les interactions gravitationnelles et électromagnétiques. Bien qu'il décéda avant d'y arriver, cette approche fait toujours l'objet de recherches, car elle aurait pu instaurer une théorie du tout factorisant et

simplifiant les formules et calculs physiques en plus de permettre de concevoir plus simplement le monde.

De la même manière que pour l'électricité, l'humain a depuis longtemps eu besoin de **transmettre des informations à distance** en usant de stratagèmes sonores comme les tambours parlants (tama), ou visuels comme des feux placés en hauteur sur des montagnes voire des tours, ainsi que les signaux de fumée pouvant en émaner. Vers la fin du 18^{ème} siècle, c'est en **1794** que **Claude Chappe** met au point des tours surplombées par des sémaphores. Elles sont constituées de trois barres reliées entre elles, supportées par un mât attaché au centre de la barre du milieu. Un mécanisme permet de les bouger pour réaliser des figures représentant des caractères et ainsi transmettre des messages à distance grâce à un **télégraphe**. Ce réseau de télécommunication avait déjà été victime de piratage par deux hommes d'affaires bordelais, dans le but de connaître les cours de la bourse de Paris avant le reste de la population. À cette période, l'idée d'un télégraphe électrique est déjà présente, mais il faudra attendre 1837 pour que naisse le télégraphe électrique commercial de Cooke et Wheatstone, suivi de peu par celui de Samuel Morse en 1840, accompagné du fameux code éponyme qui deviendra la norme des télécommunications internationales en 1865, 15 ans après les premières installations de câbles sous-marins pour relier les continents. Bien après Robert Hooke, qui inventa le **téléphone** à ficelle en 1665 ; le physicien **Antonio Meucci**, originaire d'Italie, débarque en Amérique lors de l'année 1850 dans le but de promouvoir ses inventions, dont le *Telettrofono* qu'il aurait utilisé pour parler avec sa femme immobilisée par l'arthrite. En 1871 il fonde même une entreprise et tente de protéger ses inventions, mais n'a pas assez de fonds financiers pour s'offrir un réel brevet. La personne reconnue comme l'inventeur du téléphone en aurait donc subtilisé l'idée lors d'une visite aux bureaux de Western Union Telegraph, où Meucci aurait entreposé son Télettrophone. C'est ainsi qu'**Alexander Graham Bell**, professeur de diction à l'université de Boston, mari et fils de femmes sourdes, obtient le brevet du téléphone¹⁴. Quelques heures avant, Elisha Gray déposa une demande de brevet similaire pour lequel l'invention du téléphone a été refusée. Dans la même année, il inventa un télégraphe musical, ancêtre du synthétiseur.

L'époque contemporaine est également celle des **prémices de l'affichage numérique**. En 1857 apparaît le *Tube de Geissler*, ancêtre de la lampe à décharge et des néons. Son inventeur est un physicien et mécanicien du même nom qui voulait démontrer la capacité d'un courant électrique à traverser un gaz sous basse pression pour créer un plasma dans une décharge lumineux. Entre 1869 et 1875, **William Crooke** découvre les **rayons cathodiques** lors de la réalisation de lampes thermoïoniques, ou tubes à décharge électrique portant son nom. Une vingtaine d'années plus tard, c'est **Ferdinand Braun** qui en 1897 développa un énième tube cathodique éponyme. Enfin, **Boris Rosing**, un scientifique russe d'origines néerlandaises, a utilisé des tubes cathodiques de Braun, le système de disques de l'ingénieur Paul Nipkov et des cellules photoélectriques, réalisant ainsi les fondations d'un **système de télévision mécanique** lors de l'année **1907**. Après quoi il s'est empressé de déposer une demande de brevet et

démontrer le principe de fonctionnement en détail à l'aide de schémas expliquant l'hystérésis magnétique. Ces travaux ont fini par être démocratisés par un de ses élèves et assistant de travail nommé **Vladimir Zvorykine**, qui a inventé l'**iconoscope** permettant de capter les images qui seront retransmises sur ces écrans et qui devint rapidement pionnier de la télévision aux États-Unis et en Allemagne, bien que rapidement concurrencé par **Philo Farnsworth** qui fut le premier à réaliser une **chaîne de télévision avec émission-réception**. Les tubes cathodiques ont longtemps fait partie de nos écrans de télévisions qui affichaient des images à l'aide d'une cellule photoélectrique, dont le faisceau était dévié, faisant varier le nombre d'électrons qui passait, et ainsi la luminosité ou la couleur du faisceau avant qu'il atteigne l'écran.

Entre-temps, **Louis Lumière** invente les plaques photographiques sèches instantanées prêtes à l'emploi, aussi appelées *Étiquettes bleues*, dévoilées en l'an 1881. Trois ans après, **Georges Eastman Kodak** inventa le premier **appareil photo**. S'en suivra **Georges Demeny**, qui en 1895 invente une **caméra** de prise de vues animées nommée biographe, puis viendra le Kinétographe d'Edison en 1890, et enfin, en 1895, le Cinématographe des frères lumières avec l'aide de Jules Carpentier. Pour faire la transition entre l'affichage numérique et les prémices des composants électroniques, je vous présente **André Blondel**, ingénieur polytechnique français et inventeur de l'**oscillographe bifilaire** en **1893**. C'est une machine électromécanique permettant d'étudier des courants alternatifs à l'aide de graphiques, et plus largement toute évolution d'un signal électronique dans le temps. Il inventera de nombreux autres dispositifs de mesures électriques comme un hystérésimètre et un wattmètre. L'oscilloscope succèdera à l'oscillographe et se perfectionnera avec l'ajout d'un écran cathodique, puis sa numérisation.

Le principe du semi-conducteur a été découvert en **1874** par **Karl Ferdinand Braun**, le même qui inventa les tubes cathodiques de Braun une vingtaine d'années après. Il s'est rendu compte qu'il était possible de faire passer un courant électrique dans une direction donnée à l'aide d'un fin fil de métal et d'un cristal de galène, inventant ainsi la **diode**. Au début du 20^{ème} siècle, **John Ambrose Fleming** se base sur des travaux de Thomas Edison et de Frederick Guthrie, pour appliquer leurs technologies dans le but de remplacer les détecteurs magnétiques des radios et permettre d'amplifier leurs signaux. Il invente ainsi le *kénotron*, premier **tube à vide**, **ancêtre des diodes et semi-conducteurs**. Enfin, **Lee De Forest** inventera l'Audion en **1906** avec l'ajout d'une électrode intermédiaire à une diode, permettant de convertir une variation de courant en variation de tension et de puissance. **William Eccles** renommera cette invention **triode** en 1919. Tous ont beaucoup contribué au développement des circuits électroniques et de la transmission sans fil avec la naissance de la radio.

Genèse de la programmation informatique

Maintenant que la base matérielle est en place, je vais introduire la partie théorique de la logique qui régit l'informatique. **Bolzano** est un philosophe logicien né en 1781 qui proposa son œuvre « *Théorie de la science* » pour tenter d'unifier la logique globale, pensant la chose tel un système, un tout composé de parties distinctes de ce dernier. Il a introduit le mot objet comme quelque chose qui est représenté par une idée. À l'inverse, une idée abstraite qui serait irréalisable, ne possède donc pas d'objet, et ne représente ainsi rien de concret. L'idée du « rien » n'a d'ailleurs par définition pas d'objet. Ce concept d'objet de Bolzano me fait étrangement penser à la notion de classe abstraite en programmation-objet, qui est une idée ne pouvant être instanciée en objet tant qu'elle n'est pas étendue par une autre classe non abstraite permettant quant à elle la création d'un dit objet. De même, en JavaScript, l'absence d'un objet attendu est notée `null`, qui peut être traduit en « rien ».

Entre 1844 et 1854, **George Boole**, invente une algèbre binaire éponyme. Basé sur vrai et faux, 1 et 0, l'**algèbre booléenne** formule les variables, opérations et fonctions logiques. Il a ainsi posé les bases de la logique moderne, qui sera reprise par les personnalités suivant ce paragraphe. Un peu plus d'une dizaine d'années après la naissance de l'algèbre de Boole, **Charles Sanders Peirce** en enseignera et répandra les concepts aux États-Unis à partir de 1867. Il a introduit le pragmatisme comme philosophie, mais aussi comme méthode scientifique, partant d'axiomes intuitifs pour en dégager un raisonnement logique. Sa philosophie pragmatique influença à son tour [John Dewey](#) et bien d'autres. Dans ce cadre, Peirce a élaboré une catégorisation en trois niveaux de clarté. La *priméité* est de l'ordre de la sensation permettant d'avoir une vague idée de la chose perçue. Vient ensuite la *secondéité*, en tant qu'idée singulière, reliée ou corrélée à une autre. Et enfin la *tiercéité*, qui est l'interprétation ou la représentation du concept en lui-même comme quelque chose d'habituel. Philosophe, mais surtout scientifique et logicien, il a contribué à la notation logique et il est le premier à avoir réalisé une table de vérité.¹⁵ Sa notation dite flèche de Peirce ($A \downarrow B$) est toujours utilisée aujourd'hui pour induire une porte NOR (ni A ni B) entre deux propositions logiques.

Gottlob Frege, plus jeune d'une dizaine d'années, fit correspondre plusieurs principes mathématiques à la logique et l'arithmétique, dont celui du philosophe David Hume. À cet effet, il a développé la pensée selon laquelle les principes mathématiques ont un équivalent en logique et même que la logique serait un domaine incluant les mathématiques en son sein. Ce mouvement de pensée a été nommé **logicisme**. Il a repris [la caractéristique universelle de Leibniz](#) et développé une **notation logique** qu'il a décrite dans un ouvrage nommé *Idéographie* (*Begriffsschrift* en allemand) publié en 1879. Il y fait correspondre des notations mathématiques à ses notations logiques. C'est alors que **Bertrand Russel** lui envoie un paradoxe qu'il solutionnera lui-même pour permettre à Frege d'aboutir son œuvre. Se basant potentiellement sur *How To Make Ideas Clear*¹⁶ de [Charles Sanders Peirce](#), et avec l'aide de Bertrand Russel,

Gottlob Frege fonda le mouvement de la philosophie analytique dans le but de clarifier les pensées. Russel a ensuite co-publié une œuvre de 1910 qui est au fondement de la logique mathématique. Elle est nommée *Principia Mathematica* et a été réalisée avec l'aide d'**Alfred North Whitehead** qui sera quant à lui le fondateur de la philosophie des **processus**. Enfin, les travaux des logiciens cités précédemment seront repris et aboutis par le mathématicien et professeur d'université **Ernst Schröder** dans son *Vorlesungen über die Algebra der Logik*, puis par le philosophe Ludwig Wittgenstein dans *Tractatus Logico-Philosophicus*. Pour finir, des suites d'un groupe de travail de **Xerox** voulant standardiser les caractères de code, la norme **Unicode** a petit à petit posé les bases des caractères logiques. De nos jours, nous pouvons donc écrire des propositions et des tables de vérité avec les principaux caractères logiques suivants :

$A \Rightarrow B$	$A \Leftrightarrow B$	$\neg A$	$A \wedge B$	$A \vee B$	$A \nabla B$	$A \downarrow B$	$\forall A : B$	$A \vdash B$
Implique que	Équivaut à	Négation ou inverse	A et B	A ou B	Soit A soit B	Ni A ni B	Pour tous éléments de A, on réalise l'action B	A prouve que B

F4 : Définitions de la syntaxe logique

A	B	$A \wedge B$	$\neg (A \wedge B)$	$\neg A$	$\neg B$	$(\neg A) \vee (\neg B)$
Vrai	Vrai	Vrai	Faux	Faux	Faux	Faux
Vrai	Faux	Faux	Vrai	Faux	Vrai	Vrai
Faux	Vrai	Faux	Vrai	Vrai	Faux	Vrai
Faux	Faux	Faux	Vrai	Vrai	Vrai	Vrai

F5 : Table de vérité

Dans la section « Révolutions informationnelles » du chapitre « Époque moderne », j'ai introduit Basile Bouchon, Jean-Baptiste Falcon, et Vaucanson qui ont prototypé le métier à tisser automatisé. Pour finir de tisser cette partie de l'Histoire, **Joseph Marie** Jacquard invente en **1800** le **métier à tisser Jacquard**, complètement automatisé. Cette machine inspirera **Charles Babbage**, la première personne à avoir automatisé l'édition de tables de calcul. Il a initialement travaillé avec des calculateurs humains, pour lesquels la conception de telles tables était fastidieuse à superviser et encore plus à calculer et réaliser. À cette date, il n'y avait pas d'électronique, c'est pourquoi il a œuvré à développer une machine mécanique basée sur les méthodes de calcul de l'époque. Les calculateurs humains effectuaient alors essentiellement des additions et des soustractions, sous la supervision de mathématiciens qui leur prépareraient les formules, d'après la méthode des différences finies. D'où le nom de la machine que Babbage a inventée en **1834**, « la **machine à différences** ». Ayant l'expérience du milieu, il voulait réaliser un système fiable, résilient à l'erreur. Cette dernière pouvait provenir du calcul, mais arrivait le plus souvent lors de l'impression. Il a donc fait en sorte, dès la phase de design, que sa machine prépare directement le texte en résultant pour l'impression. C'est une approche très intéressante, qui réduit les intermédiaires et automatise toute la chaîne de création. La machine est un outil qui peut guider et réduire les erreurs, ce qui à l'époque était crucial, notamment en mer, où une erreur de calcul ou d'impression sur l'almanach du navigateur pouvait mener à la

perte de tout un navire et de son équipage. De 1820 à 1830, il a visité beaucoup d'usines européennes dans le but de chercher des idées pour sa machine à différences. Il ne parvint pas à son objectif initial, mais devint un économiste des machines industrielles de son époque. À peine eut-il conçu la machine à différence, qu'il oublia la finalité de base : réaliser des tables de calcul. Il entretiendra dès lors l'idée d'une **machine capable de calculer tout ce qu'un humain pourrait lui demander, la Machine analytique**. De son vivant, seul le concept existait, elle n'a vu le jour que grâce à son fils, qui après une tentative infructueuse en 1888, revint à la charge en 1906 et réalisa une machine fonctionnelle qu'il a présentée devant l'académie royale anglaise d'astronomie, après quoi il en fit don au musée des sciences de Londres en 1910.

Le métier Jacquard et les travaux de Babbage sont donc à l'origine de la programmation. Un programme est une prévision écrite composée d'instructions pour répondre à la question « Quoi faire quand ? ». Le tout **premier programme informatique** a été imaginé par **Ada Lovelace** en **1842**, alors qu'elle a 27 ans. 10 ans auparavant, elle rencontra Charles, avec qui elle travailla, notamment sur la Machine analytique, pour laquelle elle conçut ce programme.

Héritant de tous les concepts et technologies énoncés précédemment dans cette section, **Hermann Hollerith** inventa la **mécanographie** 50 ans après qu'Ada ait écrit le premier programme informatique. Cet ingénieur américain a été recruté en tant que statisticien au Bureau de **recensement** des États-Unis. Dans la fin du 18^{ème} siècle, en 1790, le premier recensement estimait la population des États-Unis à 3.9 millions d'individus. En 1840, 28 greffiers ont travaillé à la réalisation d'une estimation de 17.1 millions. Enfin, le recensement de 1880 a nécessité 1495 greffiers qui devaient scrupuleusement pointer avec une couleur d'encre propre à chaque statistique effectuée. Avec la méthode automatisée par cartes perforées d'Hermann Hollerith dévoilée l'an **1890**, la création d'un tableau statistique a été accéléré de **10 fois par rapport à ses concurrents**, remplaçant bon nombre de greffiers. Il a construit une **machine statistique à cartes perforées** qui exploite des cartes 12x6cm regroupant les 210 cases dédiées au stockage de toutes les informations nécessaires. Son invention a permis d'effectuer le recensement auparavant manuel en seulement six ans. Par la suite, sur une idée de l'un de ses collègues, il améliora le fonctionnement de cette machine en utilisant un métier à tisser Jacquard pour mécaniser la lecture des fiches de recensement et améliorer son efficacité. Enfin, il finit par quitter l'administration et fonde la Tabulating Machine Company en 1896 qui fusionnera avec 3 autres entreprises pour fonder la Computing-Tabulating-Recording Company (CTR).

Guerres mondiales

Des suites de la période contemporaine, la relation entre le peuple et la religion a beaucoup changé. L'État étant constitué de représentants du peuple, sa relation avec l'Église a donc suivi la même logique. C'est ainsi que le 9 décembre **1905**, le régime concordataire qui définissait certains des cultes les plus pratiqués comme services publics depuis 1802, a été abrogé, séparant notamment l'État et l'Église, qui avait alors des avantages par rapport aux autres institutions religieuses. L'événement déclencheur de cette abrogation est la fameuse affaire Dreyfus, pour laquelle Emile Zola a écrit « J'accuse ! ». À partir de cette abrogation, la République garantit la liberté de conscience et de culte, mais ne finance plus aucune religion. Il faudra tout de même attendre la fin des deux guerres pour que la laïcité soit démocratisée dans les années 1960.

Guerre informationnelle et anticipation

Le conflit qui débute officiellement en 1914 est la Première Guerre mondiale. Elle a simultanément mobilisé les ressources et moyens de production de pays venant de tous les continents. Les machines de bureau deviennent électromécaniques, l'humain peut désormais rapidement transmettre des messages sur de longues distances à ses congénères à l'aide des télégraphes. Cette naissante industrie de l'information qu'est l'informatique cherchera à vendre des machines mécanographiques dans le monde entier. Elles serviront les gouvernements les plus puissants, mais aussi les pires de cette époque, pendant laquelle une entreprise en particulier se démarquera des autres. Sa genèse en est marquée par la nouvelle technologie qu'est la mécanographie, et c'est **Thomas Watson**, qui après avoir travaillé à la récente et bien portante National Cash Register company (NCR), est embauché en 1914 pour diriger le regroupement d'entreprises de la CTR, et finira par fonder l'International Business Machines Corporation (**IBM**) après la guerre, en **1924**.

Dans les **ateliers de mécanographie**, de multiples stations composées d'outils appelées machines à statistiques étaient présentes. Les opérateurs de ses machines devaient à chaque fois les reprogrammer à l'aide du tableau de connexion. Ce dernier représente un programme à l'aide de câbles qui permettent d'envoyer les données d'entrées dans les différents modules de calcul, puis de l'exécuter et récupérer le résultat affiché sur des roues totalisatrices gravées, pour enfin le recopier à la main et produire de nouveaux programmes en cartes perforées à partir de ces résultats. Ce processus était aussi long que mes phrases, c'est pourquoi en l'an **1920**, **Hollerith** présente la première machine avec module d'impression et tableaux de connexion amovibles interchangeable, permettant ainsi de rapidement changer le programme d'une machine, qui sera désormais appelée **tabulatrice**. 11 ans plus tard, en 1931, IBM sort le modèle 601. Ce calculateur électromécanique est la première machine d'IBM capable de multiplier deux nombres provenant de cartes en entrée puis de perforer la valeur résultant de ce calcul. Elle est d'ailleurs surnommée la Multiplying Punch. Les **téléscripteurs** auparavant

utilisés par les opérateurs télégraphiques ont été rendus obsolètes par le téléphone. Ils seront cependant réutilisés pour encoder des caractères qui seront convertis en binaire dans le but de travailler et **perforer des cartes à distance**. En 1935 né le réseau Telegraph Exchange ou **TÉLEX** qui permettra à deux téléscripteurs ou plus de communiquer par **ondes radio**, une technologie au cœur des avancées de l'entre-deux-guerres.

Le radar est un outil de détection et d'estimation de la distance utilisant les ondes radio (**RA**dio **D**etection **A**nd **R**anging). Sa création découle des travaux de James Clerk Maxwell et Heinrich Rudolf Hertz ainsi que ceux d'Albert Hull, l'inventeur du magnétron, une machine capable de convertir l'énergie cinétique des électrons en ondes électromagnétiques à l'aide de tubes à vide. En **1934**, la Compagnie générale de télégraphie Sans Fil française (CSF) étudie le magnétron et développe les premiers radars décimétriques. Dans le début des années 1900, le sonar était déjà bien étudié et mis en application, et pour cause, il utilise le son audible par l'oreille humaine, comme onde permettant le calcul de distances. Rudolf Kühnhold était spécialiste du domaine, mais décida en 1931 de se consacrer aux ondes électromagnétiques. C'est ainsi qu'il aboutit la création d'un système de télémètre radio, simultanément et indépendamment du dépôt de demande brevet de la CSF. L'année suivante, c'est en **1935** que **Robert Watson-Watt** sera retenu comme l'**inventeur officiel du radar**, en brevetant les indications pour l'améliorer¹⁷. Il a prouvé l'efficacité de son système en détectant des avions à 27 km puis en l'améliorant jusqu'à 100 km, mettant ainsi fin aux développements concurrents comme le sonar. Il commence alors la mise en place d'un réseau de radar côtier nommé Chain Home. En 1937 les premières stations étaient opérationnelles et elles joueront un rôle crucial en 1940 lors de la bataille d'Angleterre, durant laquelle les Britanniques ont même monté des radars sur des engins volants dans le but de détecter les bombardiers.

Pour revenir sur l'informatique, en **1936** Alan Turing invente une expérience de pensée présentant une machine nommée d'après son nom. Cette **machine de Turing** est à la base de l'informatique théorique. C'est une métaphore du fonctionnement de tout appareil de calcul, elle a posé les bases de la complexité algorithmique et de la calculabilité. Imaginez un **ruban infini** composé de cases numérotées, toutes ces cases possèdent initialement un zéro. Sur ce ruban est situé un **agent** capable de lire et écrire les symboles sur le ruban, ainsi que de se déplacer sur la case précédente ou suivante. Un **registre** viendra mémoriser la suite d'**états** allant de celui initial, jusqu'à l'actuel, le nombre d'états est limité et fini. Enfin, cet état permet grâce à une **table d'action** indiquant quoi faire à la machine, comme se déplacer avant ou après sur le ruban, ainsi que d'effectuer une lecture ou une écriture, pour enfin mémoriser le nouvel état après réalisation de l'action donnée. Si aucune action ne correspond à l'état actuel et au symbole qui vient d'être lu, alors la machine s'arrête. Avec ses quatre simples composants, il est possible, tout comme Alan l'indique dans sa publication, de créer une Machine de Turing universelle, capable de simuler le comportement de n'importe quelle autre machine de Turing. Un tel système est **Turing complet**. C'est ainsi la raison pour laquelle il est possible d'utiliser un ordinateur pour simuler une machine virtuelle, ou utiliser des jeux Turing-complete comme

Minecraft¹⁸, Terraria¹⁹, ou Factorio²⁰ pour reproduire un processeur, un affichage graphique, et ainsi une version simplifiée du jeu à l'intérieur de son propre monde virtuel. Enfin, il existe désormais un ludiciel du même nom qui guide le joueur ou l'élève pour lui apprendre l'électronique pas à pas. Ce logiciel simule sa logique et permet de reproduire des circuits complexes. Le joueur peut ainsi expérimenter et comprendre les portes logiques ou le langage assembleur qui naquit 13 ans après l'expérience de pensée de Turing.²¹

Pionnier de l'intelligence artificielle, il a proposé le **test de Turing** pour évaluer la faculté d'une machine à imiter une conversation humaine. Il la décrivit dans sa publication *Computing Machinery and Intelligence* comme un *jeu d'imitation* qui donnera le titre du film récent abordant une partie de sa vie. Enfin, alors que la série de machines allemandes **Enigma** ; qui ont été brevetées à la fin de la Première Guerre mondiale, chiffre des messages de guerre depuis des dizaines d'années en utilisant le chiffrement par substitution polyalphabétique, et dont une énième version sert en cette période de Seconde Guerre mondiale ; Alan Turing s'est consacré à son déchiffrement en se basant sur les travaux du cryptologue polonais **Marian Rejewski**, qui a étudié un **exemplaire d'un manuel** d'Enigma montrant le chiffrement d'un texte en clair à l'aide d'une clé et de son résultat chiffré. Pour arriver à ses objectifs, Alan Turing conçut une **Bombe cryptologique électromécanique** qui utilise la **force brute**, de l'anglais brute-force, c'est-à-dire qu'elle vérifie toutes les combinaisons. Cette machine sera capable de casser les codes d'Enigma et il est estimé que son déchiffrement a réduit d'au moins deux ans la durée de la Seconde Guerre mondiale.

Dans l'autre camp, l'ingénieur allemand **Zuse Konrad** commença à travailler sur une série de quatre calculateurs éponymes nommés Zuse. La première version, le Z1, était un calculateur mécanique qui sera abouti en **1938**. Il fut peu fiable, car certaines pièces ont été réalisées manuellement à la scie, dans la chambre de ses parents qui l'ont soutenu financièrement. La deuxième version en sera une version améliorée sortie en 1940, réutilisant la mémoire mécanique qui était elle très fiable, mais en remplaçant l'unité arithmétique avec l'utilisation de relais électriques. Le **Z3**, achevé en **1941**, est un calculateur électromécanique qui sera considéré comme le **premier ordinateur numérique**, il était entièrement automatique et programmable, utilisant le calcul binaire et à virgule flottante. Cette troisième version s'avérera être théoriquement Turing complet, mais ne fut pas considérée pleinement comme telle, car elle ne fournissait pas de branchements conditionnels permettant de ne pas aller à l'adresse mémoire suivante, mais vers une adresse spécifique en fonction de la satisfaction ou non d'une condition. La quatrième et dernière version de **1953** sera le **premier ordinateur commercial au monde**, avant même le Ferranti Mark I et l'UNIVAC 1. Konrad sera d'ailleurs plus considéré comme un scientifique de production et d'affaire qu'un chercheur, contrairement à Alan Turing ou Von Neumann qui conceptualisaient et publiaient avant de passer aux applications. Cela ne l'empêcha pas d'être un pionnier de l'informatique qui conçut également, de 1942 à 1946, le **premier langage de programmation** de haut niveau théorique nommé

Plankalkül, bien qu'il fut oublié de son temps, et reprit qu'en 1975 puis compilé en 2000 cinq ans après sa mort.

Pour permettre les inventions de ces deux antagonistes, les moyens de détection, de communication et de calculs se sont améliorés, de même que le stockage de l'information. En effet, l'*enregistrement magnétique* a été inauguré *sur fil* en 1900 à l'Exposition universelle de Paris, **Valdemar Poulsen** y présenta son invention nommée *télégraphone*. Inspiré par ces travaux, **Fritz Pfleumer** invente la *bande magnétique* en 1928 dans le but d'enregistrer du son, comme ça sera le cas la même année avec le *magnétophone* de **Karl Stille**, lui-même inspiré du *télégraphone*. Quelques années après, la bande magnétique sera perfectionnée en acier, et une dizaine d'années plus tard, vers 1940, en plastique. Entre-temps, l'ingénieur **Gustav Tauschek** invente une nouvelle méthode de **mémoire vive magnétique** possédant un temps de lecture de seulement quelques millisecondes contre plusieurs centaines auparavant. Cette technologie dite *mémoire tambour* sera utilisée dans la conception de machines dont le Manchester Mark I, l'ENIAC, l'IBM 650, l'IBM 701, et l'UNIVAC 1103 (version scientifique). Avant d'en arriver là, moult calculateurs seront construits pour l'effort de guerre, que ce soit pour effectuer différents calculs sur la trajectoire des projectiles, autrement dit de balistique, ou encore pour la cryptanalyse. Le Colossus Mark 1 sera par exemple utilisé pour **déchiffrer** les messages produits par les **machines de Lorenz**, utilisées par l'Allemagne nazie pour faire communiquer les hauts dirigeants de Berlin et ceux des différents corps d'armée en utilisant un chiffrement de flux en continu. La version 2 du Colossus sera quant à elle utilisée pour le débarquement de Normandie.

Howard Aiken, un doctorant de physique américain de l'université d'Harvard, a eu besoin de calculer des équations différentielles. Ne pouvant les résoudre manuellement, il s'inspira de [la Machine analytique de Charles Babbage](#) et obtint un financement d'IBM pour réaliser l'Automatic Sequence Controlled Calculator (ASCC). Datant de **1944**, c'est le **premier calculateur entièrement automatique**, il est aussi connu sous le nom de **Harvard Mark I**. Une fois lancé, il calculait pendant des heures en suivant le programme qui lui était donné à l'aide de cartes perforées. Ces avancées lui ont valu une couverture médiatique dans la presse, voyant cette machine comme un robot intelligent pouvant trouver toutes les réponses mathématiques. Ce calculateur ne possédait pourtant pas encore de branchement conditionnel, alors que la Machine analytique de Babbage, bien que jamais construite, stipulait déjà cette fonctionnalité. Ses programmes étaient alors très longs et il possédait des Relay Switches qui pouvaient casser. L'ASCC a permis la réalisation de tables et la réalisation de premières expérimentations notamment par [Grace Hopper](#).

La **théorie des jeux**, comme son nom l'indique, étudie les jeux et plus notamment le comportement des joueurs ainsi que leurs décisions. Elle se base sur l'hypothèse qu'ils effectuent toujours un choix rationnel en fonction des connaissances à leur disposition, et ce dans le but de maximiser leurs gains et minimiser leurs pertes. La plus ancienne démonstration de la sorte que je connaisse est le Pari de Blaise Pascal, déclarant que si Dieu n'existe pas, alors

y croire comme le renier n'a pas d'importance. Mais dans l'hypothèse où il existe, alors il vaut mieux y croire pour ne pas finir en enfer. Le choix rationnel permettant les gains potentiels maximum serait ainsi selon lui de croire en Dieu. Avant même les deux guerres mondiales, **Ernst Zermelo**, un assistant de Max Planck, a jeté les bases de la formalisation mathématique des jeux à somme nulle avec un article « *Sur une application de la théorie des ensembles à la théorie du jeu d'échecs* » publié en 1913. Enfin, Émile Borel, pionnier de la théorie des mesures, des probabilités, et des jeux, publiera un article sur ce dernier domaine dans les *Comptes rendus hebdomadaires des séances de l'Académie des sciences* du 1er juillet 1921¹⁶. Puis en 1940 il publia un guide de « *Théorie mathématique du bridge à la portée de tous* » coécrit avec l'aide d'André Chéron.

Ces théories ont été appliquées aux jeux économiques, **Antoine Augustin Cournot** fut pionnier en 1838 dans ses « *Recherches sur les principes mathématiques de la théorie des richesses* » dans lesquelles il analysa un duopole et vint à être l'un des premiers à déclarer qu'il existe un équilibre entre l'offre et la demande. Il faudra attendre 1944 pour qu'Oskar Morgenstern et John von Neumann formalisent ce domaine de recherche dans leur ouvrage « *Théorie des jeux et du comportement économique* ». Enfin, **John Forbes Nash**, aboutira les travaux de Cournot et inventera lui aussi un équilibre à son nom, le fameux équilibre de Nash. De la même manière que le radar a permis d'anticiper la venue d'ennemis, la théorie des jeux permet d'analyser et de prédire les actions qui sont les plus intéressantes ou probables d'arriver.

Vannevar Bush était chercheur en électrotechnique au MIT, il a participé à la réalisation de machines résolvant des équations différentielles ainsi que d'autres, destinés au traitement automatique de microfilms de stockage analogique. Titulaire d'une chaire à l'ancêtre de la NASA, alors nommé NACA, il fut conseiller scientifique de Roosevelt et parmi les chercheurs du projet Manhattan de 1945. En juillet de cette même année, il anticipe l'avenir de l'information dans l'article *As we may think*²² paru dans le mensuel The Atlantic. Le même mois, il assistera aux premiers essais de bombe nucléaire, qui sont arrivés à peine un mois avant les bombardements atomiques d'Hiroshima et Nagasaki, la Seconde Guerre mondiale se termine alors. Cet article est un appel aux scientifiques, qui ont jusque là mis leurs savoirs au service de la guerre, semblant à juste titre arriver à sa fin ; de désormais partager la connaissance, trouver des moyens d'inscrire une information de manière quasi permanente, et de la retrouver plus facilement. Il déplore notamment que les travaux de certaines personnes, comme le concept des lois de la génétique de Mendel, aient été perdus pendant toute une génération du fait que sa publication n'ait à l'époque pas atteint un public capable de reprendre ses travaux. Il indique que contrairement à [Babbage](#) et [Leibniz](#), la situation technologique et économique nous permet désormais de stocker et gérer l'information scientifique afin qu'elle soit conservée, transmise et surtout consultée, à l'aide de machines de moins en moins chères et de plus en plus fiables, dont ils sont tous deux pionniers. Il parlera des avancées de la photographie, de la capture vidéo, et de la compression utilisée sur les microfilms qu'il a longuement étudiée. Il introduit alors la possibilité qu'à l'avenir, une machine puisse enregistrer

et convertir la parole d'un auteur en texte, comme le faisaient alors les sténographes ; ajoutant qu'une machine des laboratoires Bell nommée *Voice Operating Demonstrator* (Voder), présentée lors d'une exposition internationale, était actuellement capable de convertir du texte en une voix humaine synthétique ; et qu'une autre était capable de l'opposé, c'est-à-dire de convertir une voix en caractères phonétiques (à la manière d'une sténotype), appelée quant à elle Vocoder. Vannevar Bush projette alors que les auteurs du futur pourront utiliser des machines non plus analogiques, mais électroniques, utilisant des signaux radio pour communiquer à distance à une vitesse éclair, traitant des informations bien plus variées, avec des usages plus versatiles, conjointement à une possibilité de suppression et de réécriture.

D'un point de vue de l'information, il énonça plusieurs concepts, dont la sélection, avec pour exemple les numéros de téléphone permettant de sélectionner un destinataire parmi des millions de stations possibles ; mais aussi l'indexation, indiquant que l'information est stockée à un endroit unique, suivant un chemin de groupes et sous-groupes se finissant généralement par une liste triée alphabétiquement. Il introduit alors la notion d'association et de liens entre les informations, formant une toile, dans laquelle l'utilisateur pourrait se balader, gardant un historique du chemin qu'il a emprunté pour passer d'un article ou d'une idée à l'autre. Pour illustrer ses propos, il finit par introduire un outil fictif de son invention. Le **memex** permettrait à tout un chacun de stocker ses livres, enregistrements, communications, suppléant ainsi sa mémoire. Il l'image comme un bureau, composé d'écrans translucides, d'un clavier, de leviers et de boutons, potentiellement utilisables à distance ; et évoque alors la possibilité d'acheter du contenu sous forme de microfilms à insérer dans la machine. Il imagina des interactions utilisateurs, comme l'utilisation de leviers pour faire défiler les pages d'un document, ou d'un stylet pour ajouter des notes de marges. Ces concepts seront sources d'inspiration pour la construction du [World Wide Web](#) avec l'hypertexte, les sélecteurs [HTML](#), et même Wikipédia dont l'usage du memex décrit en réalité parfaitement le fonctionnement, formant comme il l'indique « *une nouvelle forme d'encyclopédie formée de chemins associatifs interconnectés* ».

Lors du sous-chapitre précédent, j'ai brièvement parlé de calculateurs capables d'effectuer des calculs balistiques. L'anticipation du résultat d'un tir a toujours été la préoccupation des militaires. Dans l'art de la guerre, il est tout aussi important de toucher sa cible que d'éviter une balle perdue qui pourrait toucher des civils. La Moore School est l'actuelle université de Pennsylvanie. Initialement dédiée à la formation d'ingénieur en électronique, elle vit le jour en 1923. À la même période apparaît le Ballistic Research Laboratory, qui comme son nom l'indique, est un établissement de recherche en balistique. Ces deux établissements auront une importance cruciale dans le développement de l'informatique. En **1937**, un professeur de mathématiques et de physique nommé **John Vincent Atanasoff** entreprend, avec l'aide de **Clifford Berry**, un de ses étudiants diplômés, la création d'une machine à calculer électronique. Deux ans plus tard, en 1939 une machine rudimentaire est alors existante, mais incomplète, elle sera nommée **Atanasoff Berry Computer** (ABC). Pendant son développement, ils ont apporté des idées pionnières et innovantes comme l'arithmétique

binaire, l'utilisation d'éléments de commutation électronique, et la parallélisation. La construction de cette machine est d'ailleurs la première à intégrer des tubes à vide et elle sera considérée comme la **première unité arithmétique logique** qui compose aujourd'hui les processeurs. En 1940, Atanasoff et Berry présentent leur machine à **John William Mauchly**, qui a voulu apporter une solution à la réalisation de tables de tir, permettant de régler correctement une arme balistique pour atteindre sa cible, située à une certaine distance. Idéalement le tireur doit réaliser une unique tentative précise, plutôt que de devoir faire un essai approximatif et réajuster en fonction de la trajectoire observée comme c'était le cas jusque là. Après 5 ans de conception, c'est en 1942 que la machine est testée avec succès, elle sera dès lors utilisée pour trouver des solutions à des équations linéaires. Par la suite, Mauchly s'allie à **John Adam Presper Eckert**, et à partir de juin 1943 ils travaillent sur la réalisation de l'**ENIAC** à la Moore School, afin de pouvoir calculer des trajectoires balistiques. Il sera quant à lui le premier ordinateur électronique Turing complet. Même s'il a été pensé par 2 hommes, l'**ENIAC** a été programmé par 6 femmes, qui ont documenté, codé et branché des millions de câbles. Des suites de la réalisation de l'**ENIAC**, Mauchly et Eckert formeront la Eckert-Mauchly Computer Corporation en 1946. Malheureusement, la mort de leur investisseur principal lors d'un trajet en avion les poussera à revendre leur entreprise à Remington. Elle sera ensuite rachetée par Sperry Corporation après fusion avec la Burroughs Corporation en 1986, et deviendra Unisys, 100 ans après sa création.

Guerre froide

La Seconde Guerre mondiale prend fin et c'est officiellement l'époque contemporaine pour les Anglo-saxons. C'est l'avènement du nucléaire, de l'ère de l'informatique, mais aussi de la décolonisation et de la guerre froide. Les besoins de calculs sont plus importants que jamais. Aéronefs, nucléaire militaire ou civil, ainsi que la gestion de l'énergie sont au cœur des réflexions et stratégies étatiques. L'arrivée du calcul et de la simulation informatique offrent un meilleur contrôle, ainsi qu'une prise de décision plus efficace, permettant de gérer de grandes structures, qu'elles soient civiles, militaires, ferroviaires, aériennes, ou autres. De grands projets à forte complexité en découlent, ils amènent à un besoin de communication multimédia mondiale pour que les gens puissent s'organiser et faire coopérer des spécialistes de tous les domaines concernés. Internet viendra résoudre ce problème à partir de 1970, amenant à la période de mondialisation intense que nous vivons toujours en 2024.

Genèse de l'informatique moderne

Beaucoup d'ordinateurs ont été réalisés en **1945**, au crépuscule des deux guerres mondiales. Parmi eux, l'**EDVAC**, l'**ENIAC**, le **SSEC** succédant au **ASCC**, et le **Z4**. **Lors de ses travaux sur l'EDVAC, John Von Neumann** a décrit une architecture dans laquelle le stockage possède le code ainsi que la donnée à traiter sur le même support, permettant désormais aux instructions du programme d'être traitées comme de la donnée, et donc de pouvoir être lues et réécrites.

Cette **architecture dite de Von Neumann** sera réutilisée par Alan Turing et tous les ordinateurs à partir de cette date. Enfin, Mauchly et Eckert modifieront l'ENIAC pour implémenter son concept avec l'aide de Von Neumann lui-même.

Le tube de Williams-Kilburn, breveté à partir de fin 1946 par Frederic Calland Williams et Tom Kilburn, utilise les tubes cathodiques de [Boris Rosing](#) pour enregistrer des données binaires. La **Small-Scale Experimental Machine (SSEM)**, premier ordinateur à architecture de von Neumann, les utilisait pour sa mémoire vive. Le test de cette nouvelle technologie sur la SSEM, aussi appelé Manchester Baby étant concluant en **1948**, la production d'un autre ordinateur fut amorcée. Commence ainsi la réalisation du Manchester Mark I, qui voit le jour seulement un an après, en **1949**. Cette machine fut décrite comme un cerveau électronique par la presse. Des neuroscientifiques s'y intéressent rapidement, à une période où naît la cybernétique. En effet, quelques années avant, plus précisément en **1947**, le mathématicien **Norbert Wiener** publie *La cybernétique*, il y introduit ce qui deviendra un domaine d'étude non seulement des mécanismes d'informations, mais également des systèmes complexes ainsi que leurs analogies entre les organismes vivants et les machines. Mettant notamment en avant la capacité d'un système à se réguler lui-même, et les moyens mis en œuvre pour atteindre une finalité, soit respectivement la *rétroaction*, et la *téléologie*, deux notions aussi importantes que présentes dans ce domaine. La cybernétique trouve son origine étymologique du grec *kubernêtikê* qui signifie gouverner dans le sens de diriger un navire. L'une des premières machines pourvues de rétroaction est le régulateur à boules de James Watt de 1788, qui permettait de réguler la vitesse de rotation d'une machine à vapeur. Ce mouvement de pensée et la doctrine qui y est liée rallieront de nombreux scientifiques, penseurs et mathématicien, dont John Von Neumann.

Warren Weaver est principalement connu comme un des pionniers de la traduction automatique, il a assisté Claude Shannon dans le développement de son article nommé *A Mathematical Theory of Communication*, paru en **1948**. Il y introduit la théorie de l'information qui traite de comment coder l'information, si possible en la compressant voire la chiffrant en utilisant la cryptographie, et ce afin d'optimiser les transmissions d'informations au travers d'un moyen de communication, qu'il soit oral, téléphonique, électronique, ou autre. Son approche étant mathématique, il utilise beaucoup de notions de probabilité et d'incertitude, représentant tout ce qui pourrait interférer avec la transmission du message, ou encore les chances mêmes que l'information qu'il véhicule soit véridique. C'est à partir de ce moment que sera employé le terme de **bits**, contraction de **binary digits** ou nombre binaire en français, unité de base en informatique. Par souci de quantification hexadécimale, 8bits forment un octet.

Jusqu'alors, les instructions des premiers langages de programmation étaient donc écrites en binaire, difficiles à lire et interpréter par un humain. **Maurice Wilkes** et **David Wheeler**, qui travaillèrent en **1949** sur l'Electronic Delay Storage Automatic Calculator (**EDSAC**), et le Binary Automatic Computer (**BINAC**), ont alors récupéré l'idée d'une notation utilisée par Von Neuman et Herman Goldstine. Pour exemple, un ingénieur pouvait indiquer la notation simplifiée **A 25 S** sur papier pour décrire l'instruction d'ajouter dans l'emplacement mémoire

de numéro 25 l'entier court (Short type) pouvant stocker un nombre entier entre -32 767 et 32 767. Cette instruction était alors récupérée par un technicien qui le convertissait en binaire, obtenant `1110000000110010`. Une fois inscrite en binaire dans le système, il était compliqué de « debugger » et voir où était située une potentielle erreur. Wilkes, s'est rendu compte que l'ordinateur, capable de manipuler les nombres et les lettres de l'alphabet grâce au binaire, pouvait théoriquement lui-même stocker `A 25 S` et le convertir en `1110000000110010`. Il décida alors de confier la rédaction d'un programme capable de convertir les instructions symboliques d'un téléscripteur en un code binaire à Wheeler, qui passait alors son doctorat. Cette avancée formidable a permis les fondations des langages de deuxième génération comme **l'assembleur**, dont les instructions sont simples, lisibles par l'humain, mais spécifiques à une certaine famille de processeur dont elle permet de manipuler les commandes principales à l'aide de mot clé mnémonique. Ils ont ainsi automatisé une tâche tout en mettant en exergue un autre problème. Les programmes en résultant devaient s'exécuter correctement et produire les résultats escomptés. Pour cela des techniciens devaient relire attentivement le programme. Or la provenance des erreurs semblait venir de la réimplémentation de la même fonctionnalité de manières différentes. La solution, également proposée par Von Neuman et Herman Goldstine a donc été de mettre en place une bibliothèque de petits programmes spécifiques alors appelée routine. Ainsi, les « développeurs » de l'époque pouvaient juste y faire appel plutôt que de les réinventer à chaque fois. L'idée de factorisation et de code réutilisable est alors née, permettant d'améliorer la productivité et la fiabilité.

Le premier ordinateur commercial sort en **1951** et s'appelle l'Universal Automatic Computer I ou **UNIVAC**. C'est alors une filiale de Remington Rand. Sa mémoire était gérée par une bande magnétique qui défilait rapidement avec l'aide de servomoteurs. Cette dernière était initialement en plastique, mais des suites de sa déformation, la bande était souvent rompue. Son matériau de réalisation a donc été changé pour du métal résistant, bien qu'il fallut du travail pour pouvoir écrire et lire dessus. Pour la réalisation de cet ordinateur, **Grace Hopper**, docteur en mathématiques, officier de la marine, et informaticienne américaine ; crée le **premier compilateur** en **1951**, le Arithmetic Language ou **A-0** system. Il y a eu plusieurs versions jusqu'à arriver au **B-0** Flow-matic, développé entre 1955 et 1959. Comme tous les compilateurs, il prend un code composé d'instructions textuelles et le convertit en instructions binaires exécutables par la machine, ce que le A-0 réalisait déjà en établissant des liens entre les programmes et sous-programmes ainsi que leurs paramètres. En **1957**, **John Backus** ingénieur d'IBM, invente le Formula Translator ou **FORTRAN**, un langage de programmation de haut niveau de troisième génération qui produira des programmes binaires aussi performants que ceux des programmeurs expérimentés. Il a initialement conçu ce langage comme une solution à un problème. Il a constaté que plus de la moitié des coûts d'un centre informatique étaient dus aux salaires des développeurs, qui passent jusqu'à la moitié de leurs temps à tester et debugger plutôt qu'à produire du code. À cela s'ajoute le fait que les machines devenant de moins en moins onéreuses, ce pourcentage ne ferait alors qu'augmenter si rien n'était fait. C'est en présentant cette logique à son supérieur qu'il obtint l'autorisation de travailler sur ce projet qui

n'a eu à son début que peu de reconnaissance, considéré comme un projet de recherche tentant de réaliser l'impossible, et n'ayant aucune garantie de résultats. Lorsque Backus et son équipe ont présenté le projet à des développeurs, ils étaient tous sceptiques. Leurs attentes étaient hautes et le FORTRAN encore approximatif et inabouti. Cela faisait en effet quelques années que les salariés d'IBM entendaient parler de « programmation automatique ». Ce concept permet d'automatiser la programmation bas niveau en utilisant un langage de plus haut niveau d'abstraction pour ainsi factoriser les instructions à la manière de programmes comme les assembleurs, interpréteurs, compilateurs et générateurs de programmes. Ce type de programme s'exécutant sur d'autres programmes voire dans certains cas sur eux même est du domaine de la **métaprogrammation**. La réalisation comme la popularité du FORTRAN ont donc pris du temps à se mettre en place. Son développement et son adoption ont été accélérés par l'édition de manuels simplifiés comme le *Programmer's Primer* de Grace E. Mitch. Par la suite, le **LISP** sera inventé en **1958** par **John Mc Carthy**, et **Grace Hopper** inventera le **COBOL**, un langage de haut niveau qui verra le jour en **1959**. Dans ce langage, les instructions étaient moins techniques et plus lisibles, donnant l'illusion aux décideurs que n'importe qui pourrait comprendre les programmes qui en découlent, pour preuve, celle pour calculer le salaire net d'un employé à partir de son salaire brut était ainsi :

```
SUBSTR TAX FROM GROSS-PAY GIVING NET-PAY
```

Des langages de plus haut niveau se sont ensuite développés. La quatrième génération et ses abstractions encore plus hautes permettent de facilement interconnecter et manipuler des outils logiciels tiers, comme une base de données, une interface graphique, ou un serveur web. La plupart des langages de la troisième génération se sont dotés de ces fonctionnalités et ont évolué vers cette quatrième génération. Ce genre de langage est désormais assez haut niveau pour permettre un développement rapide de programmes dans un domaine spécifique. C'est entre autres la raison pour laquelle Python s'est spécialisé dans le traitement de la donnée et JavaScript dans les interactions utilisateurs. Enfin, une cinquième génération de langages apparaîtra. Dans ce paradigme, l'ordinateur trouve lui-même les solutions pour nous. Très lié à l'intelligence artificielle, l'exemple le plus concret en est Prolog.

Temps réel

Initialement chiffré à deux ans de développement et 200'000 dollars, pour finalement atteindre les 8 millions de dollars pour huit ans ; un projet américain top secret nommé *Project Whirlwind* visait à simuler l'espace aérien des États-Unis. Bien qu'il ait coûté une fortune, cet ordinateur, constitué initialement de tubes à vide, puis par la suite l'un des premiers à avoir utilisé de la mémoire à tores de ferrites, a surtout permis des sorties en **temps réel**. Il influença cette technologie et bâtit les fondations de l'industrie informatique jusqu'à aujourd'hui. Cette machine devait simuler un avion, et même potentiellement n'importe quel avion sur le territoire

américain. Pour que les opérateurs de cette simulation aient une expérience réaliste, il est évident que les affichages et donc les valeurs en sorties de l'ordinateur doivent toujours être à jour, nécessitant logiquement du temps réel. C'est pourquoi **Jay W. Forrester**, la personne en charge du projet, fit la rencontre de **Perry O. Crawford Jr.** Ce pionnier de l'informatique était alors l'un des premiers, si ce n'est le premier, à avoir réalisé et promu les mérites des ordinateurs numériques au profit des analogues, avant même que de tels ordinateurs ne soient construits. En effet le premier ordinateur numérique est l'ENIAC de 1945. Crawford et Forrester l'ont découvert à la fin de la guerre, lorsque les projets secret défense ont alors été révélés, à l'occasion d'une conférence du MIT sur les Techniques de Calcul Avancés. Mais alors que les budgets sont revus à la baisse à cause de la fin de la guerre, le projet Whirlwind est mis en danger, et sa faisabilité à court terme remise en cause. Arrive alors la guerre froide, avec la menace de bombe nucléaire, missiles et bombardiers russes, capables d'arriver aux États-Unis en passant par le pôle nord.

En 1950, George Valley occupe le rôle de professeur de physique au MIT et de membre du conseil scientifique de l'armée de l'air américaine. Il fit un rapport déclarant que le système de défense antiaérienne américain était catastrophique. Les informations du système de radar étaient à ce moment traité manuellement avant d'être relayées à un centre de contrôle et de commande. Les problèmes de communication et de couverture radar rendaient impossible une alerte rapide en cas de tentative d'attaque par le pôle nord. Valley fut par conséquent responsable du projet qui sera nommé **Semi-Automatic Group Environment (SAGE)**, avec pour but de solutionner ces problèmes. Pour la réalisation de ce projet, il chercha alors un moyen informatique pour améliorer la récupération, les communications et le traitement des données radar. Bien qu'il ait eu écho du Projet Whirlwind en mal, il décida de se renseigner par lui-même. Il se rendit vite compte des progrès impressionnants qui avaient été réalisés, et qui par chance avaient enfin permis l'exécution de programmes de tests. Il relança donc le projet avec les fonds nécessaires, rendant le système opérationnel l'année suivante, en 1951. À partir de ce moment, le Whirlwind fusionne avec SAGE. Un réseau de 23 centres de directions est alors distribué dans tout le pays, et des opérateurs commencent à analyser les engins volants, qu'ils soient civils ou militaires, alliés ou ennemis, le tout sur des écrans cathodiques. Ce système en temps réel donnera lieu, dans les années suivantes, au projet SABRE de réservation aérienne, jusqu'alors manuel et fastidieux, qui sera lui-même généralisé à bien d'autres systèmes comme le bancaire avec l'apparition des cartes de crédits, distributeurs automatiques, codes barres, et plus tard paiements en ligne. Plus récemment, les États-Unis ont déclaré que les ovnis sont un problème de sécurité nationale, cela laisse penser qu'un tel système est sûrement encore dans les tuyaux. Enfin, Perry Crawford avait prédit que le temps réel ne permettrait pas seulement la simulation d'un avion, mais la visualisation en temps réel de tout le trafic aérien, comme c'est le cas aujourd'hui en source ouverte. Les circuits imprimés, écrans cathodiques et mémoire à tores seront grâce à ce projet, et aux intervenants comme IBM, Burroughs, ou Bell ; rapidement démocratisés et vendus aux particuliers dans les

années 1960. Le développement des écrans provoquera la naissance de nouvelles interfaces plus pratiques.

Âge d'or d'IBM

IBM était initialement une entreprise qui réalisait des calculateurs comme les modèles IBM 602 et 603 en **1946**, le 604 deux ans après en 1948, ou encore l'IBM 407 et son imprimante capable d'imprimer 150 lignes par minutes. Vint ensuite l'IBM Card-Programmed electronic Calculator, rival historique de l'EDSAC et du BINAC, qui fait son apparition la même année que le 407, en **1949**. Mais en 1951, la livraison de l'UNIVAC I fit grandement peur à IBM qui craignait de perdre une partie importante du marché civil, bien que cela ne fut pas le cas. Leur réponse a été de confier la réalisation d'un ordinateur à **Nathaniel Rochester**, qui a travaillé sur nombre de technologies comme le radar, les éléments arithmétiques du projet Whirlwind I, la reconnaissance de schémas, l'intelligence artificielle, et la théorie de l'information qu'il a dirigé. Grâce à son aide, IBM dévoila ainsi le modèle **IBM 701** en **1952**, premier ordinateur scientifique produit en masse, aussi connu sous le nom de Defense Calculator. Il possédait une mémoire tambour, et utilisait le premier langage assembleur conçu par Rochester. L'année d'après en **1953**, John Backus, alors ingénieur d'IBM, a créé le Speedcoding. C'est un interpréteur qui a été le premier langage de programmation de haut niveau créé pour un ordinateur IBM. Bien qu'il s'appelle Speedcoding, et qu'il ait facilité le développement en fournissant des instructions pour les fonctions mathématiques, le fait que ce soit un interpréteur le rendait particulièrement lent, 10 à 20 fois plus que du code machine. De plus, il prenait 30% de la mémoire à lui seul. Malgré cela, il a permis la prise en charge des nombres à virgule flottante sur les ordinateurs IBM 701. **L'IBM 702** lui succédera la même année, il est le premier ordinateur muni de dérouleurs de bandes magnétiques. Il était moins puissant que le 701 ou le ERA 1103, mais avait, contrairement à eux, les civils pour cible commerciale. Contrairement à l'UNIVAC, sa mémoire n'était pas constituée de ligne à retard analogique, mais elle était électrostatique grâce à des tubes de William, deux fois plus rapide et plus fiable. De plus, contrairement au monolithe qu'était l'UNIVAC, sa conception étant modulaire et composée de boîtes reliées entre elles. Le 702 pouvait ainsi être transporté dans la majorité des ascenseurs. Après son annonce, IBM arrive à réaliser 50 commandes en 10 mois. Toujours en **1953**, est développée la **mémoire à tores magnétiques** qui a révolutionné la mémoire vive pendant une vingtaine d'années. Constitué d'anneaux de ferrite polarisés, correspondant à un bit de 0 ou 1 en fonction du sens du champ magnétique, tous traversés par des fils conducteurs. Cette technologie se base sur l'hystérésis pour faire passer un courant dans un fil afin de lire, ou d'écrire la valeur des tores si le courant est assez élevé. Elle poussa IBM à rééditer ses modèles 701 et 702 avec ce type de mémoire, en plus de la création de nouveaux modèles l'utilisant comme le 704 et 705 EDPM. L'année suivante, sortira l'**IBM 650** de **1954** également pourvu de ce type de mémoire. Ce fut le modèle le plus populaire des années 50, et pour cause, il

coûtait seulement un quart du prix d'un 701. Il a été vendu à de nombreuses universités, créant toute une génération de programmeurs. Cela a fait de lui le premier ordinateur produit en masse dans le monde et la première machine d'IBM à réaliser un profit significatif.

Créé aux États-Unis en **1955**, le **TRAnsistor DIgital Computer** des laboratoires Bell sera le premier ordinateur à transistors. Le grand public retiendra quant à lui l'IBM 1401, fabriqué entre **1959** et 1965. Il fut l'ordinateur à transistor le plus vendu, notamment grâce à la politique marketing d'IBM qui a adopté une vision globale, dans l'objectif de prendre en compte le maximum d'utilisateurs et donc clients potentiels. Cette politique avait déjà fait ses preuves avec le modèle 650. IBM a ainsi créé plus qu'un ordinateur, un système complet, avec une imprimante pouvant imprimer 600 lignes par minutes, principale raison d'achat ayant fait son succès. En plus de cela, avec ses transistors flambants neufs et bien plus fiables que les tubes à vide, il rendit obsolètes les ordinateurs en étant muni, en même temps que les machines électromécaniques qui étaient encore utilisées par soucis monétaires. Par la même occasion, IBM invente le **Report Program Generator (RPG)**, un langage de programmation de haut niveau facile à comprendre, permettant aux comptables et techniciens de tabulatrices de se « reconverter » en utilisant des concepts qu'ils connaissent et utilisaient déjà sur les panneaux de contrôles, où ils branchaient des câbles pour implémenter les entrées, calculs, et sorties. Ils pouvaient ainsi programmer sans apprendre tous les arcanes de l'assembleur, du COBOL, ou du FORTRAN. Malgré cela, beaucoup de clients préféraient demander les logiciels directement à IBM, qui avait un monopole si important qu'ils pouvaient se permettre de les livrer « gratuitement » avec les machines qu'ils louaient.

Alors que Control Data Corporation (CDC) s'attaque au marché scientifique, fournissant des ordinateurs avec un meilleur rapport performance / prix, les plaçant troisièmes derrière UNIVAC et le leader IBM ; General Electric annonce 3 ordinateurs, un bas, un moyen et un haut de gamme ; la Radio Corporation of America (RCA) et Honeywell décident quant à eux de réaliser un ordinateur compatible avec le 1401 et d'autres machines IBM ; cette dernière se rend compte que ses concurrents ont décelé une opportunité et qu'ils doivent réagir à cette menace mettant en danger leur domination sur le marché. IBM commence alors la conception d'une gamme d'ordinateurs compatibles entre eux, ayant la même architecture, et capables d'exécuter les mêmes logiciels ; ce qui n'était à ce moment pas le cas, posant problème autant à ses clients qu'à IBM même. Ces machines alors très spécialisées n'offraient pas une grande interopérabilité ou possibilité de changements de logiciel comme de matériel et périphériques, voire de chaîne de production, nécessitant des experts de la machine en question. Contrairement aux machines à cartes perforées, les ordinateurs n'avaient donc à ce moment pas de normes et standards, nécessitant parfois de recoder un même programme sur N machines différentes, un processus très onéreux. IBM décide alors de concevoir le **System/360** en **1962**. Il n'a pas été économe pour autant, 5 billions de dollars, c'est plus que le budget du projet Manhattan. Cela provoqua des différends monumentaux dans leurs équipes, au point où les dirigeants ont été obligés d'isoler celle en charge du 360. Niveau marketing ils auraient pu tout annoncer d'un coup, mais ont décidé d'annoncer machine par machine, de manière à avoir une adoption

progressive comme cela avait été le cas pour le passage des machines électromécaniques aux ordinateurs dans les années 50. En 1963 Honeywell lance donc le modèle 200 compatible IBM 1401, modèle qu'IBM décida de rendre progressivement obsolète, pour ensuite lancer la production du System/360 composé de 6 ordinateurs et 44 périphériques. En **1964**, sa communication marketing se fera dans 63 villes des États-Unis et 14 pays étrangers. S'en suivent plus de 5 billions de chiffre d'affaires et 30 ans de croissance. Cet « *ordinateur a été fait par IBM et a fait IBM* ». Il avait pourtant bien des défauts, il ne supportait pas le temps partagé, permettant d'être utilisé par plusieurs utilisateurs en même temps. Mais la stratégie et la communication ont su compenser le côté technique suffisant.

Émergence de l'Intelligence Artificielle

Comme je l'ai évoqué précédemment, depuis la parution de *La cybernétique* de Norbert Wiener qui a contribué à l'analogie entre êtres vivants et machines, la capacité des ordinateurs à prendre de la donnée en entrée, la traiter et obtenir un résultat en sortie, a rapidement amené la presse à décrire cette technologique comme des machines pensantes, avec une vitesse de traitement bien plus rapide que celle d'un cerveau humain. L'idée d'une intelligence artificielle était donc présente depuis les premières machines capables de suppléer ou remplacer les calculateurs humains. Mais la démocratisation du terme et la naissance de l'intelligence artificielle comme domaine de recherche sont considérées comme ayant eu lieu durant l'été **1956**, avec la **conférence de Dartmouth** qui a eu lieu au collège du même nom. **John Mc Carthy**, du haut de ses 28 ans, y est à ce moment professeur de mathématique assistant. Après avoir cherché un financement pour l'organisation de cet événement, il parvint à rallier de grands noms du domaine comme Marvin Minsky, Claude Shannon, Nathaniel Rochester, lui même, et bien d'autres, pour un total de 11 conférenciers.

L'année qui suit, c'est en **1957** que le psychologue **Frank Rosenblatt** publie son livre *Le Perceptron*. Initialement prévu pour être une machine, l'idée en a été abstraite à une représentation mathématique et informatique d'un neurone biologique. Le perceptron représente désormais un algorithme d'apprentissage supervisé qui prend des entrées, généralement quantitatives, c'est-à-dire des nombres, et retourne toujours un booléen en sortie (1 ou 0, vrai ou faux). Le perceptron simple n'est constitué que d'une seule couche qui ne peut classer que les ensembles de valeurs ou, disons de points, clairement séparable par une ligne entre elles. Pour les problèmes de classification de plus de deux ensembles séparés par des lignes courbes, il faudra attendre **1965** pour qu'**Alexey Ivakhnenko**, étende ce simple concept en chaînant plusieurs couches de perceptrons, donnant ainsi naissance à l'apprentissage profond (**deep learning**). À partir de cette date, d'autres procédés comme la descente de gradient ou sa rétropropagation ont permis d'améliorer cette technologie et de répondre à des problèmes de plus en plus complexes.

Toujours en **1965**, deux informaticiens, un médecin et un chimiste inventent **Dendral**. Ce programme permet d'identifier des structures moléculaires en se basant sur les connaissances d'experts et des techniques d'analyse telles que la spectrométrie de masse. Le fait que ce programme interactif puisse imiter le raisonnement d'un expert du domaine fait qu'il est considéré comme le premier de la famille d'intelligences artificielles dites « *systèmes experts* ». Dendral fut par la suite utilisé comme outil d'aide à la décision. Pour identifier des molécules complexes, le nombre de possibilités est important et donc difficile à définir par un humain, il offrait ainsi l'avantage de rapidement donner les différentes combinaisons réalisables, en se basant sur les règles de la chimie et la masse moléculaire obtenue par spectrométrie.

L'année suivante, le programme **ELIZA** sera révélé. C'est une intelligence artificielle conversationnelle créée en **1966** ayant pour but de simuler un psychothérapeute, notamment en tournant des affirmations de l'utilisateur en question. Ce logiciel prenait les phrases de l'utilisateur en entrée, essayait d'identifier des tournures de phrases communes et potentiellement en extraire un ou plusieurs mots à réutiliser dans la réponse. Il était programmé pour utiliser des phrases comme « Je comprends. » ou « Je suis désolé d'entendre que tu es [insérer un adjectif négatif] » dans le but de donner une illusion d'empathie qui poussera certains utilisateurs à ne pas faire la différence avec une conversation humaine, faisant ainsi d'ELIZA l'un des premiers robots conversationnels à passer le [test de Turing](#).

Entre-temps, de la même manière que la presse a comparé les machines au cerveau humain, **Hilary Putnam**, philosophe de l'esprit, du langage et des sciences a établi le **computationnalisme** en **1960**. À cette époque, l'approche en vogue était le comportementalisme qui assumait que les individus réagissaient aux stimuli extérieurs uniquement en fonction de réflexes ou de leurs expériences passées. Le computationnalisme quant à lui consiste à penser l'ensemble des capacités du cerveau humain, c'est-à-dire l'esprit, comme un système complexe de traitement de l'information. Ce système biologique est en effet capable de dérouler des étapes et passer par différents états, à la manière d'un algorithme ou d'un automate, convertissant une entrée, en l'occurrence une pensée ou des sensations, en une réaction en sortie. Cette théorie est englobée dans le cognitivisme qui ne se cantonne pas qu'au traitement de l'information, mais inclue notamment tout le système de perception, d'attention, et même de motricité ou d'affects. Ces deux théories aboutiront au connexionnisme des années 1980 qui aborde les phénomènes mentaux en les modélisant avec des réseaux de neurones issus du simple perceptron. L'intelligence artificielle est alors lancée autant en tant que technologie que de concept et se développera rapidement. Deep Blue vaincra le champion du monde d'échec Gary Kasparov en **1997**, IBM Watson remporte le jeu télévisé Jeopardy en **2011**, et Alphago devient le meilleur joueur de Go au monde en 2016.

Du circuit électronique au microprocesseur

Le microprocesseur est un circuit intégré, tous deux sont des circuits imprimés, dont l'origine est située quelques dizaines d'années avant leur démocratisation. C'est pourquoi j'ai besoin de faire un léger retour en arrière, avant même les deux guerres mondiales, à la période où apparaissent les semi-conducteurs. En effet, les premiers prototypes de **circuits imprimés** apparaissent au début du 20^{ème} siècle, composés de métal conducteurs appliqués sur une carte faite d'un matériau isolant aussi appelé *diélectrique*, ils ont été des premières tentatives de circuits électroniques complexes. En 1903 Albert Hanson dépose un premier brevet²³ pour coller des fils conducteurs sur un papier de paraffine dans le but d'améliorer les branchements téléphoniques, posant les bases de cette technologie. Une vingtaine d'années plus tard, Charles Ducas obtint un brevet datant de 1925, pour un procédé utilisant la galvanoplastie électrolytique afin de graver un circuit électronique grâce à un pochoir, décrivant par la même occasion la possibilité de réaliser des cartes ayant plusieurs couches de circuits conducteurs.²⁴ Deux ans après, César Parolini réalisera de même un brevet sur l'impression de motifs adhésifs en poudre de cuivre sur un diélectrique, suivant un procédé de Ducas et une idée de Thomas Edison.²⁵ Enfin, c'est en 1943 que Paul Eisler a recours à un énième brevet de gravure de motif conducteur, utilisant cette fois-ci des feuilles de cuivre plaquées sur une base isolante servant de support au circuit électrique.²⁶

En 1958, **Jack Kilby** est ingénieur chez **Texas Instruments**. Il y réalise le **premier circuit intégré**, qui sera plus tard connu sous le nom de puce électronique, en reliant manuellement deux transistors grâce à du germanium. Bien qu'il fut longtemps oublié, Kilby finira par obtenir le prix Nobel de physique en 2000. Revenons en 1958, **Robert Norton Noyce** travaille alors à **Fairchild Semiconductor** qu'il a cofondé un an avant. Il y réalise indépendamment la même découverte et popularise les circuits intégrés en **silicium** à la base de l'électronique toujours utilisée aujourd'hui, éclipsant ainsi Kilby. Suite à un désaccord au sein des dirigeants de Fairchild, Noyce décide avec Gordon Earle Moore, lui-même cofondateur, de quitter cette entreprise pour fonder Intel. De la même manière que Backus avait constaté que le matériel devenait de plus en plus fiable et de moins en moins cher pour de meilleures performances, Gordon inventa par la suite la **loi de Moore** postulant que les semi-conducteurs auraient une progression linéaire de leurs performances, qui doublerait ainsi tout les deux ans. Entre-temps, des améliorations comme les trous traversants plaqués, initialement brevetés en 1961 par la Hazeltine Corporation, ont permis des cartes plus fiables et complexes. Aujourd'hui, les cartes électroniques peuvent avoir une centaine de couches de circuits imprimés.

Pour réaliser des processeurs, il faut des **transistors**. Le principe de ces derniers est connu depuis un brevet datant de l'année 1925²⁷, soumis par un doctorant dont la thèse a été tutorée par Max Planck, j'ai nommé **Julius Edgar Lilienfeld**. Il y décrit la méthode et un appareil capable de contrôler la tension, c'est-à-dire le voltage, d'un courant électrique. Cependant, la réalisation effective du transistor se fera en **1947** par des chercheurs américains du Laboratoire Bell. Un an après, des Français réalisent indépendamment un composant similaire nommé

transistron. La presse spécialisée leur donnait l'avantage technique d'être plus stables et résistants, mais le gouvernement français était alors focalisé sur les recherches nucléaires.²⁸ Avant les années 70, les composants des processeurs étaient trop volumineux pour tenir sur les circuits imprimés fraîchement créés une dizaine d'années avant. Mais en **1969** Federico Faggin et Marcian Hoff, respectivement ingénieur et physicien d'Intel, réalisent cet exploit et inventent le **microprocesseur**.

Course à l'espace

Spoutnik 1 fut lancé et mis en orbite par les Russes en **1957**. C'est le **premier satellite d'origine humaine** et il n'a pas eu d'autre rôle que celui d'envoyer un signal sonore par ondes radio, un simple « bip-bip », l'espace de seulement 3 mois. Pourtant, dans le contexte de la guerre froide, cette prouesse russe a marqué les esprits, notamment américains, qui y voyaient la menace d'une attaque nucléaire par voie spatiale. D'autant plus qu'en 1961, les Russes enchériront avec la mission Vostok 1, faisant de Youri Gagarine le premier homme à aller dans l'espace. L'Amérique et ses services spatiaux se sont alors mis pour objectif d'être le premier peuple humain à aller sur la lune. Dix ans après la mise en orbite de Spoutnik 1, suite à un incendie lors des tests censés précéder son lancement, la mission **Apollo 1** de **1967** est malheureusement un échec et la lune paraît de nouveau inaccessible aux Américains. Deux ans après, sous la direction de Charles Stark Draper ; Margaret Hamilton, qui a travaillé sur le projet SAGE, réalise avec l'équipe qu'elle supervise, le programme qui sera utilisé par le premier ordinateur à circuit intégré, le Apollo Guidance Computer (AGC). Ce programme écrit en Assembleur a été numérisé et rendu disponible sur GitHub. C'est notamment grâce à ce logiciel et aux calculs de trajectoires et de fenêtres de lancement réalisés et vérifiés par Katherine Johnson, que le 21 juillet 1969, Apollo 11 fut finalement une réussite. Le bien connu Niels Armstrong deviendra ainsi le premier homme à marcher sur la lune et les États-Unis ont rattrapé leur retard dans la course à l'espace jusqu'alors menée par l'URSS.

Bilan passé

À travers mon étude sur l'histoire de l'informatique, j'ai remarqué que les inventions de savants et créateurs proviennent rarement de nulle part. Leurs origines sont souvent des (re)découvertes de principes existants, que leurs inventeurs compilent et réutilisent à leur manière, améliorant les idées de leurs prédécesseurs et réalisant de meilleurs outils à force d'itérations innovantes. Des technologies comme le radar ou la pile électrique ont trouvé leur inspiration dans des observations de la nature, notamment des animaux avec les chauves-souris qui se guident à l'aide de l'écholocalisation, ou les anguilles électriques capables de produire des courants allant jusqu'à 860V et 2A. Les personnes pionnières dans leur domaine sont généralement suivies par foule de passionnés, car les idées naissent d'une combinaison d'autres, déjà existantes dans le

contexte mondial à un moment donné. Il est en effet impossible de savoir qui a eu l'idée d'une chose pour la première fois, l'Histoire ne retiendra naturellement que la première personne à avoir popularisé le terme ou l'idée, parfois à travers la commercialisation d'un produit, d'autre fois grâce à une publication scientifique ou un brevet. Certaines informations plus ésotériques ne se transmettront que par la parole, conservant ainsi un secret relatif.

Bien que les prémices de la gestion de l'information automatisée viennent du 17^{ème} siècle lors de l'Époque moderne, période avec un besoin grandissant de calcul ; l'informatique naquit après la révolution industrielle, au 19^{ème} siècle lors de l'Époque contemporaine, et ne commença à réellement émerger qu'après la Seconde Guerre mondiale qui marque le début d'une nouvelle ère de progrès technologiques, accompagnée d'une forte augmentation de la population humaine causée par le baby-boom. Les machines à calculer ne datent pourtant pas d'aujourd'hui, l'Anticythère remonte tout de même à -150 ! Cependant, le premier programme informatique a été imaginé en 1842, et il a fallu attendre 1950 pour voir apparaître l'assembleur et les premiers langages de haut niveau.

Pratiques actuelles

Jusqu'alors, l'informatique était essentiellement réservée aux domaines privés comme le militaire, le spatial et l'administration des États. À partir des années 70, l'informatique arrive dans le quotidien des particuliers les plus aisés et finit par être dans une grande majorité des foyers. Internet commence à se développer petit à petit. L'humanité assiste alors à la popularisation et à la mondialisation de l'information ainsi que de ses systèmes de gestions.

Genèse d'internet

Un an après le lancement russe de Spoutnik, c'est en 1958 que le président américain de l'époque, Dwight David Eisenhower, déclare le lancement d'un projet initialement nommé **Advanced Research Projects Agency**, soit ARPA ou DARPA après que le nom du projet ait été préfixé par le mot Défense. Le premier projet de cette agence fut de faire face aux besoins d'amélioration de la couverture et de la précision des réseaux de détection terrestres comme le radar, et de permettre à des sous-marins de la marine américaine de se géolocaliser précisément pour ainsi mieux calculer leurs tirs. La solution technique de ce projet du début des années 60 est un système de navigation par satellite nommé Transit, précurseur du projet **NAVSTAR** lancé en 1973, et opérationnel une vingtaine d'années plus tard. Ce projet donnera finalement le système **GPS**, popularisé grâce à l'autorisation de libre diffusion de ses signaux par Bill Clinton en 2000, permettant un usage civil avec une précision à 10 mètres près. Mais revenons brièvement en **1969**, année où le premier réseau informatique nommé **ARPANET** fit son apparition, notamment grâce à l'aide de scientifiques ayant travaillé sur le projet SAGE

comme Joseph Licklider. Leur expérience sur le projet SAGE a apporté des connaissances en temps réel et partagé, ainsi qu'en interconnexions d'ordinateurs en réseau, permettant de mener à bien la réalisation de ce qui deviendra internet.

Alors qu'elle travaille à l'institut de recherche de Stanford, **Elisabeth Jocelyn Feinler** ou « **Jake** » est recrutée par Douglas Engelbart pour travailler un guide des ressources d'ARPANET en vue d'une conférence. En **1974**, elle fut la principale instigatrice du **Network Information Center** (NIC), organisme de gestion du réseau informatique ARPANET. Le NIC est à l'origine du système de nom de domaine (**DNS**), permettant de ne retenir qu'un mot comme `google.fr` qui sera traduit en adresse IP (`172.253.122.94`) ; ou encore du **WHOIS**, l'annuaire permettant de savoir à qui appartient un nom de domaine. Toujours en 1974 et pour les besoins d'ARPANET, Bob Kahn et Vinton Cerfs inventent le protocole **TCP/IP** pour palier au manque de gestion d'erreur du Network Control Protocol, matériellement restreint aux premiers routeurs, les Interface Message Processor.

Quelques années avant, c'est en 1971 que le français Louis Pouzin, travaillant alors à l'Institut National de Recherche en Informatique et en Automatique (INRIA), recrute Hubert Zimmermann pour développer le concept de paquet de données transmises sur un réseau, plus connu sous le nom de datagramme. Ce concept technologique a fortement contribué au développement d'ARPANET. L'année suivante, **Louis Pouzin** commence le développement de **Cyclades**, le premier réseau informatique français. Par la suite cet ARPANET français sera abandonné au profit de **Transpac**, sorti en **1978**, et utilisé par le **Minitel**. La même année aura lieu la finalisation du standard représentant les niveaux de communication réseau, j'ai nommé **le modèle OSI de Charles Bachman**, à qui les développeurs doivent l'existence des Systèmes de Gestion de Bases de Données (**SGBD**). Enfin, depuis sa création en 1980, la version 4 du protocole IP est le protocole standard, et ce malgré le fait que le nombre d'adresses IPv4 soit récemment arrivé à ses limites, auxquelles IPv6 pallie désormais grâce à une cohabitation des deux versions sur le réseau. C'est aussi lors des années 1980, dans le cadre de son travail au CERN, que **Tim Berners-Lee** réalise le projet **ENQUIRE**, avec pour objectif de réaliser un système compatible avec différents réseaux capables de gérer du contenu hypertexte éditable directement depuis le serveur, comportant des hyperliens et des données venant d'une base. Tim Berners Lee est l'inventeur du web sémantique, c'est-à-dire un réseau de données traitées et fournies par les machines dans le but d'aider ses utilisateurs à assimiler, créer, et partager des connaissances. La Française d'origine sénégalaise Rose Dieng Kuntz travailla longtemps sur ce sujet pour lequel elle édita un ouvrage nommé *Méthodes et outils pour la gestion des connaissances*. Dans les années **1990**, le **World Wide Web** succède à ENQUIRE. C'est alors que le navigateur internet **Netscape** apparaît, suivi du World Wide Web Consortium ou **W3C** en **1994**. Également fondé par Tim Berners-Lee, cet organisme de standardisation va favoriser l'interopérabilité des technologies du web telles que [HTML](#), [CSS](#), [PNG](#), [SVG](#), et j'en passe. Avec ces solides bases qui se mettent en place, **Ward Cunningham** créa le premier **wiki** en **1995**. Il est toujours disponible tel quel sur internet²⁹.

Interfaces graphiques et périphériques de pointage

Inspiré par [le memex décrit par Vannevar Bush](#), **Ivan Sutherland** inventa un logiciel d'édition d'images vectorielles pour la présentation de sa thèse de doctorat au MIT de **1963**. L'écran affichait directement les formes dessinées par l'utilisateur à l'aide d'un crayon optique. Pour l'époque, le fait d'avoir une interface graphique et de pouvoir dessiner directement dessus en ayant le rendu en temps réel était une prouesse technique révolutionnaire. Il appellera son œuvre **Sketchpad** et obtiendra à cet effet bien des honneurs et récompenses, dont un Turing Award en 1988. Cela inspirera à **Douglas Engelbart** la réalisation du **oN-Line System (NLS)**, le premier système d'exploitation multi-utilisateur à utiliser la **souris** qu'il a lui-même inventé en **1963**. Ce système a introduit les liens hypertextes, écrans à balayage, les interfaces graphiques et leurs fenêtres. Ce projet sera présenté vers la fin de l'année 1968 lors d'une conférence de 90 minutes décrite comme la « Mère de toutes les démonstrations » (Mother of all demos), mais il sera rapidement abandonné à cause de sa difficulté d'apprentissage. Beaucoup de contributeurs du NLS décident alors d'aller travailler pour Xerox, qui en **1970**, met en place le **Palo Alto Research Center**, plus connu sous le nom de **Xerox PARC**.

Ce laboratoire d'études travailla sur les premières interfaces graphiques et éditeurs permettant de modifier un document directement depuis son rendu visuel, c'est une technologie qui est communément appelée « What You See Is What You Get » ou **WYSIWYG**. Le premier logiciel à implémenter cette technologie et permettre le repositionnement du curseur ainsi que la sélection de texte à l'aide d'une souris fut l'éditeur de documents Xerox Bravo sorti en 1974, peu après l'ordinateur Xerox Alto, qui fut révélé le 1er mars 1973. Cet ordinateur est le premier à fournir une interface graphique, et ainsi à ne plus être dépendant de la ligne de commande. De surcroît, il introduira les métaphores de bureau, documents, classeurs, et corbeille, familiers à tout employé de bureau. Le couple interface graphique et pointeur favorisa grandement l'adoption et la démocratisation des ordinateurs, cela deviendra un standard omniprésent dans nos interactions avec les ordinateurs.

Smalltalk est un langage de programmation objet créé en 1972 au Xerox Parc par Alan Kay. C'est le premier à avoir un environnement de développement complètement graphique, indépendant de la ligne de commande ou d'un éditeur de texte. Il fut intégré au Xerox DoRADO, héritier du Xerox Alto, avec pour objectif d'être un système bien plus performant. Plus tard, en 1996, Smalltalk deviendra Squeak qui deviendra lui-même l'actuel Scratch utilisé pour introduire à la programmation de manière visuelle. Ils sont tous inspirés de Logo, qui est à la fois une famille de langages de programmation et une philosophie de l'éducation. C'est le résultat de travaux sur le cognitivisme promu par [Marvin Minsky](#) et les travaux sur l'éducation de Jean Piaget promus par Seymour Pagert.

Ordinateurs personnels et jeux

L'usage des ordinateurs devenant plus accessible, le marché a commencé à produire des ordinateurs dédiés au grand public. Parmi eux, le Programma 101 de 1965, l'Amstrad en 1968, l'Altair 8800 en 1975, ainsi que le Commodore PET et l'Apple II parus en 1977 ; mais aussi le Minitel, dont j'ai parlé précédemment, l'Osborne 1, l'IBM PC 5150 et le Xerox Star^[w4], tous disponibles au tout début des années 1980 ; ou encore le Macintosh 128K sortit quelques années après en **1984**, avec des icônes en skeuomorphisme, imitant des objets réels que l'ordinateur a remplacés en abstrayant leurs fonctions, telle la disquette qui représente une sauvegarde, ou la corbeille indiquant l'endroit où se retrouvent les fichiers qui vont être détruits.

À cette même période apparaissent la mémoire flash et le stockage optique sur disque. L'informatique devenant plus accessible et fun, des jeux vidéos ont rapidement fait leur apparition. Les premiers furent essentiellement des jeux simples comme le morpion, implémenté par le premier jeu vidéo connu daté en 1950. Bertie the Brain, permettait pour la première fois d'affronter « l'intelligence artificielle ». Il est rapidement suivi d'une implémentation du jeu de Nim. À partir de 20 unités, que deux joueurs en retirent chacun leurs tours une, deux ou trois, et ce jusqu'à ce que le joueur qui doit retirer la dernière unité ne perde. Après quoi furent inventés deux premiers jeux vidéos multijoueurs, *Tennis for Two* (qui inspira *Pong*), et *Spacewar!*. Tous furent réalisés avant que les interfaces graphiques ou l'ordinateur ne deviennent grand public, leur l'influence a été retentissante.

En 1970, John Horton Conway, célèbre mathématicien ayant travaillé sur la théorie des jeux, et des nœuds³⁰ ; invente un jeu pas comme les autres qu'il appela « jeu de la vie ». C'est en réalité un automate cellulaire dont les règles sont très simples, il y a une grille constituée de cases blanches et noires, si une case blanche a 3 cases noires voisines, elle deviendra elle-même noire au tour d'après, si une case noire n'a pas 3 ou 4 cases noires voisines, elle meurt et devient blanche au tour d'après. Ces règles permettent des patterns inattendus qui n'étaient pas prévus initialement par son créateur. Mais comme Leonard de Vinci l'aurai déclaré : « La simplicité est l'ultime sophistication ». À force de tests et d'itérations, il est possible d'observer des structures stables, d'autres qui se déplacent, et bien d'autres. Les utilisateurs ont fini par les utiliser pour recréer une machine de Turing à l'intérieur de ce jeu³¹, et même réimplémenter le jeu de la vie, dans le jeu de la vie.³² Enfin, les bien connus Atari Pong en 1972, Pac man en 1980, ou Tetris en 1984 ont fait leur apparition aux prémices de l'industrie du jeu vidéo.

Logiciel libre, open source, et cadriciels

À partir de 1975, des amateurs d'électroniques et d'informatique se rejoignent et forment le Homebrew Computer Club. Ils éditent des newsletters et organisent des présentations ou expositions de nouveautés, matérielles comme logiciel. Parmi eux, les fameux Steve Wozniak et Jobs, cofondateurs d'Apple, mais aussi **Richard Stallman**, qui annonce le projet **GNU** le 27 septembre **1983**. En **1985** il fonde la Free Software Foundation, où Free ne signifie pas (seulement) gratuit dans le sens de prix, mais (également) dans le sens de liberté. Stallman inventa aussi la General Public Licence en 1989, plus connu sous le nom de GPL. Peu avant, Andrew S. Tanenbaum invente Minix, un système d'exploitation UNIX avec une architecture 16 bit. Inspiré par ce dernier, Linus Torvald a par la suite créé **Linux** en **1991**, avec un nouveau noyau (kernel) nommé lineage, sous licence GPL. La coopération entre Stallman et Torvald donnera GNU/Linux qui est une suite embarquée d'éditeurs comme bash, emacs, la GNU C Library, le compilateur GCC, et plein d'autres. Le langage C était alors grandement utilisé.

Aujourd'hui, beaucoup d'entreprises utilisent du code open source. Certains utilisent même à tort des logiciels GPL sans savoir que cette licence est contaminante, faisant que leur logiciel devrait légalement être libre de facto. D'autres licences permettent aux entreprises d'utiliser des bibliothèques pour un usage commercial. Certains projets en sources ouvertes comme Symfony ont donné naissance à une multitude de business, à commencer par le leur. SensioLabs, l'entreprise derrière Symfony, organise des conférences et formations payantes, en plus de sponsoriser certaines entreprises expertes dans cette technologie, dont eux-mêmes. Symfony est un outil logiciel facilitant la réalisation d'application web. En l'utilisant, je me suis rendu compte qu'il ne permet pas certaines choses et l'installation d'un de leurs plugins React m'a causé défaut. J'ai donc eu, par la force des choses, à contacter la personne chargée de la documentation de Symphony. Ce fut une expérience riche durant laquelle j'ai découvert un environnement qui m'est familier, bien que je n'ai pas souvent réalisé de l'open source avant. Un pipeline qui sort des erreurs à des lignes qui n'existent pas sur ma version, et qui malgré cela permet d'assurer la qualité de la documentation rédigée en DOCTOR-RST, à l'aide de tests de compilation.

L'open source permet à tout le monde de bénéficier d'outils logiciels standardisés dont il est possible de vérifier le code afin de s'assurer qu'il ne comporte pas de faille, et éventuellement de remonter tous les problèmes aux développeurs, voire de contribuer directement à les résoudre en proposant des solutions ou améliorations en tous genres. Cela permet, sous réserve de respecter les licences, de modifier le logiciel pour en faire sa propre version. Le logiciel libre, quant à lui, est un pied de nez aux logiciels propriétaires. Grâce à la standardisation et à l'ouverture du code source des logiciels, les développeurs ont vu apparaître de multiples langages et cadriciels de ces derniers. Par exemple le langage CSS a permis de diversifier l'apparence des sites web, et ses cadriciels ou thèmes comme Foundation ou Bootstrap ont régénéralisé et harmonisé l'affichage des sites, réduisant la créativité au profit de l'accessibilité. Tout cadre vient limiter la liberté, généralement pour le bien commun.

Dans la continuité des éditeurs de textes open sources comme vim ou emacs, l'environnement de développement le plus utilisé de nos jours est open source. Suite à Atom publié par GitHub le 26 février 2014, Visual Studio Code est révélé un peu plus d'un an après par Microsoft, qui a en quelque sorte refait Atom, avec des performances bien meilleures et un écosystème plus actif. Après avoir englouti Atom, Microsoft a même fini par racheter GitHub.

Chiffrements actuels

En partie grâce au code en source ouverte, il existe désormais une multitude d'implémentations des d'algorithmes de chiffrement. Tout développeur back-end digne de ce nom se doit de vérifier laquelle est utilisée et si elle n'a pas été compromise. La dernière version de Message Digest nommée MD5 a été inventée en 1991 et fut partiellement compromise seulement 5 ans plus tard en 1996. En 2004, des chercheurs chinois l'ont complètement craqué. Malgré les recommandations de changer de méthode de chiffrement, MD5 est parfois utilisé à tort pour chiffrer des mots de passe, alors que sa seule utilisation devrait être de chiffrer deux fichiers et comparer leurs signatures pour s'assurer qu'ils sont les mêmes, lors d'un téléchargement par exemple. Depuis, la version 6 de Message Digest est apparue, mais sa fonction reste cantonnée à la réalisation d'empreinte numérique de fichiers. Le développement de MD5 et MD6 ont été réalisés ou supervisés par Rivest Ronald. En 1977, en compagnie de Shamir Adi, et Adleman Leonard, est décrite pour la première fois la méthode de chiffrement asymétrique RSA, dont le brevet du MIT datant de 1983 a expiré en 2000. Ce chiffrement est utilisé dans les paiements électroniques et de manière générale pour chiffrer des données sur internet. En tant que développeurs, nous l'utilisons souvent pour générer des clés permettant de se connecter à distance et de manière sécurisée à des terminaux en ligne de commande (Secure SHell ou SSH). Bien qu'il existe des propositions d'attaque pour casser RSA, une clé suffisamment grande (≥ 2048 bits) est largement suffisante de nos jours. Le principe du chiffrement asymétrique est simple à comprendre. Il faut générer une paire de clés cryptographiques. Une clé de chiffrement publique à partager pour que nos interlocuteurs puissent chiffrer leurs messages. Et une clé de déchiffrement privée qu'il convient de garder secrète pour déchiffrer les messages qui seront chiffrés par notre clé publique. Enfin, la révolution récente en termes de chiffrement utilise les courbes elliptiques. Ce genre de courbes ne sont pas communes, il peut y avoir plusieurs valeurs en ordonnée pour une seule valeur en abscisse. Une droite passant par n'importe quel autre point qu'un sommet de cette courbe la croisera donc au moins 2 fois. Les propriétés mathématiques de ce genre de courbe rendent le chiffrement plus sûr. Des algorithmes de chiffrement utilisent ce genre de courbes, lorsque je le peux, et comme conseillé par Gitlab, je génère mes clés en utilisant l'algorithme EdDSA. Ce dernier se base sur les courbes d'Edwards tordues dévoilées en 2008.

Décisions et gestion de projet

Toute entreprise ou groupe d'humains travaillant sur un même projet doit en avoir une gestion planifiée. Cela se met en place avec divers outils, mais repose essentiellement sur une bonne veille, suivie de décisions adaptées permettant finalement de découper le projet en tâches simples déléguables à des techniciens. Il convient de préparer suffisamment le travail pour qu'il soit réalisable facilement. Plus l'information sera claire et précise, moins il y aura besoin d'aller-retour entre les pôles. Il n'est pas souhaitable d'éviter ces allers-retours pour autant. Il faut travailler main dans la main et communiquer efficacement pour se répartir les tâches en fonction des responsabilités de chacun. Quand [Charles Babbage](#) visita des établissements de gestion comme les maisons d'échanges bancaires ou le bureau du cadastre en charge de réaliser des cartes à jour pour établir une taxation, il était fasciné par la division du travail qui s'y opérait. Le directeur du cadastre était Gaspard De Prony, qui réalisa les « *Grandes tables de logarithmes* » en seulement deux ans grâce à ses lectures des travaux d'[Adam Smith](#), notamment de son œuvre « *Recherches sur la nature et les causes de la richesse des nations* ». Ce dernier a été le premier à détailler le principe de division du travail, utilisé depuis la nuit des temps pour optimiser les temps de formations du personnel et de réalisation d'un produit.

Le marché du logiciel s'est vite organisé pour diviser le travail en pôles. Comme évoqué en préface, l'équipe back corresponds aux développeurs en charge de la gestion de la donnée et de la logique de traitement. Les développeurs front quant à eux sont en charge de récupérer de la donnée depuis le serveur et l'afficher selon les maquettes résultantes de l'équipe conceptrice d'interface. Cette équipe est généralement aussi séparée en deux, l'équipe expérience utilisateur (UX) met en place le squelette de l'interface (wireframe) et les interactions possibles par l'utilisateur. Les concepteurs d'interface utilisateur (UI) représentent quant à eux la direction artistique donnée à ce squelette en l'agencant proportionnellement et en lui donnant des couleurs. Enfin viennent parfois les rédacteurs qui choisissent et testent les meilleures phrases possibles pour attirer le lecteur et avoir un meilleur référencement sur les moteurs de recherche.

On comprend donc facilement que le management moderne a eu besoin de pouvoir former et guider efficacement ses employés. Pour cela il a fallu trouver des outils et moyens mnémotechniques afin de facilement analyser les facteurs influant sur l'activité d'une entreprise. Une des plus anciennes techniques est la carte heuristique, plus connue sous le nom de carte mentale. Née de l'arbre de Porphyre, un philosophe néoplatonicien du 3^{ème} siècle, elle représente visuellement le cheminement de la pensée et l'interconnexions des idées afin d'en extraire les toutes les informations importantes. Prendre une décision n'est pas chose aisée, il est parfois difficile de mesurer les tenants et aboutissants de nos choix. C'est pourquoi il est important de prendre du recul pour lister les différentes possibilités et acteurs de la situation. Beaucoup d'outils comme la carte mentale heuristique analysent d'abord un panel d'informations variées, parmi eux l'analyse Politique, Économique, Socioculturel, Technologique, Écologique, et Légal nommée PESTEL, permet de se poser des questions sur

les influences externes, et par la suite réaliser une analyse plus fine des parties prenantes. Elle peut également permettre de trouver des opportunités ou menaces à mettre dans une matrice multicritère ou dans un SWOT (Strength Weakness Opportunity Threat). Une fois que nous avons analysé l'environnement du projet, ses avantages et inconvénients, internes et externes ; nous pouvons être amenés à choisir entre plusieurs fournisseurs ou solutions. Pour cela il est possible d'utiliser une matrice multicritère. Chaque été, avec des amis, nous l'utilisons pour choisir dans quel AirBnb nous allons partir. Nous y renseignons une note pour le prix, l'emplacement géographique, la qualité du logement, et cetera... et obtenons ainsi un score moyen permettant de déterminer où nous partirons en vacances.

Après les décisions préliminaires, il est préférable de définir clairement le projet. S'il s'agit d'une entreprise, il convient de réaliser un plan d'affaires. Autrement et de manière générale, il est possible de réaliser une simple charte de projet. Son rédacteur y indique les dates de début et de fin, les principaux acteurs du projet et son nom, suivi de son périmètre et de ses objectifs. Un objectif est dit SMART s'il est **spécifique, mesurable, atteignable, réaliste et temporel**. Une fois que c'est fait, nous sommes en mesure de réaliser une planification grossière comportant les étapes clés en réalisant une feuille de route (roadmap). Il convient ensuite d'affiner cette planification en découpant les grandes parties d'un projet en petites tâches. Certaines dépendent d'autres et doivent par conséquent être réalisées à postériori. Hormis l'interdépendance des tâches, il est parfois compliqué de savoir quoi faire en premier parmi une grande liste de tâches. Pour les planifier et ordonnancer, la matrice d'Eisenhower est un outil élémentaire. Très proche des matrices de faisabilité et de priorisation, elle permet de classer les tâches importantes et/ou urgentes, celles qui ne le sont pas, ainsi qu'à qui les confier. Il est également possible de peser les tâches pour les répartir dans des livrables de poids similaires et réaliser les plus « lourdes » en premier afin qu'elles soient fonctionnelles à la date limite.

Lorsque les prérequis ainsi que les conditions de réalisation des tâches et leurs priorités sont clarifiés, il est possible de définir un diagramme de GANTT pour les planifier précisément et visualiser leur déroulé attendu dans le temps sur un calendrier. Il ne reste plus qu'à suivre cette planification au mieux, bien que les estimations ne peuvent que rarement anticiper les risques imprévus. Si un problème apparaît lors de la réalisation d'une tâche, une méthode simple et efficace nommée « 5 Why » indique que se poser 5 fois d'affilée la question « Pourquoi ? » permet généralement de trouver la source d'un problème. Dans le même genre, nous avons le diagramme d'Ishikawa qui permet de schématiser les différentes causes possibles d'un événement ou effet. Inspiré des quatre causes d'Aristote, il est généralement construit en analysant les 5 M : **M**atière première constituante, **M**atériel utilisé, **M**éthode employée, **M**ain-d'œuvre intervenant, et **M**ilieu dans lequel l'événement apparaît. Ce n'est pas la seule méthode qui est inspirée d'Aristote, QQQQCCP est un acronyme compliqué résumant une méthode interrogative empirique simple qui trouve ses origines dans L'Éthique à Nicomaque.³³ Elle correspond aux 7 questions fondamentales à se poser pour faire de tout d'une situation : Qui, Quoi, Où, Quand, Comment, Combien, Pourquoi.

Enfin, lors de ma lecture de Lean Startup, deux concepts de gestion des problèmes m'ont particulièrement marqué. Dès l'apparition du moindre problème, il convient d'arrêter toute la chaîne de production. Cela évite ainsi de contaminer la suite de la chaîne, et de propager une erreur jusqu'à l'étape finale de livraison. La seconde idée est que « lorsqu'une erreur survient, la faute nous revient de l'avoir rendue si facile à commettre ». Ici « nous » représente les managers. Autrement dit, lorsqu'un travailleur faillit à sa tâche, il ne faut pas remettre en question sa responsabilité ou sa compétence, mais se demander pourquoi l'environnement n'a pas permis la réussite de l'action et rendu la situation d'échec si probable qu'elle s'est produite. Lorsque je commets une erreur, j'en assume la responsabilité, mais je me justifie souvent pour identifier les causes du problème et clarifier la situation afin d'éviter que cela ne se reproduise.

Pour la réalisation du projet en lui-même, il y a trois principales méthodologies de gestion, la cascade (Waterfall), le cycle en V, et Scrum. Le modèle en cascade est très linéaire, il part d'un début pour arriver à une fin en passant par des étapes très définies. Le cycle en V fait correspondre les étapes préliminaires de définition du projet avec des étapes de tests et de validation qui ont lieu après l'implémentation. Scrum quant à lui est une méthode flexible plus appropriée aux projets complexes remplis d'imprévus. À la manière des autres méthodes, elle découpe le projet en différentes parties nommé « sprints », mais elle le fait tout au long du projet, et non pas de manière définitive à l'initialisation de ce dernier. Dans le cas de la réalisation d'un simple site web vitrine, le cycle en V est généralement la méthode utilisée. La complexité étant moindre, il est envisageable de dérouler sa réalisation de manière linéaire. Bien qu'il soit théoriquement possible de l'améliorer à posteriori, c'est généralement comme un magazine. Une fois qu'il est imprimé, il n'est plus modifié, mais il est éventuellement possible d'en faire une autre version quelque temps plus tard. Pour une application ou un logiciel, l'utilisation est plus dynamique, l'utilisateur a plus d'interactions possibles avec le produit et par conséquent plus d'envies, de demandes, de besoins, et de potentiellement de problèmes. C'est pourquoi la méthode Scrum est plus appropriée dans ce cas. L'équipe peut, à toute étape du projet, recueillir les retours clients, définir un objectif, réfléchir aux manières d'y répondre, puis prototyper la meilleure solution retenue, pour enfin la tester. Ce processus est très proche du *design thinking*. Cette méthode de création et de gestion de l'innovation permet de rapidement obtenir un produit minimal viable (MVP) sur lequel itérer.

Aujourd'hui le logiciel vient suppléer ces techniques de management. Le recueil et l'analyse de données permettent d'apporter une aide à la décision. La gestion des entreprises, autrefois réalisée de tête ou sur papier avec de multiples dossiers, est désormais centralisée dans des progiciels de gestion intégrés (Enterprise Resource Planning ou ERP en anglais). L'apprentissage est facilité par des ludiciels comme Adibou, Lapin Malin, Duolingo, Gymglish, ou flexboxfroggy pour le CSS, codingame pour les langages de programmation, hackthebox pour la cybersécurité, et j'en passe. Malgré cela, l'industrie du jeu vidéo représente plus de 50% des 650 milliards de valeur marchande du logiciel dans le monde. Elle occulte celle des logiciels éducatifs en ligne, et ce à raison de 396 milliards de dollars de marché contre 166. Les progiciels quant à eux ne représentent que 50 milliards (sources : Statista 2022).

De mon expérience, j'ai pu constater la différence entre prestation de service informatique et édition de logiciel. Le modèle économique d'une entreprise influence énormément les comportements de ses salariés. Les contraintes de temps et de budget ne sont pas les mêmes, les projets s'organisent différemment. Les prestataires vendent du temps, et ont donc naturellement tendance à avoir des délais plus courts et par conséquent plus de pression. Cela nécessite une équipe commerciale solide capable de décrocher assez de projets pour faire travailler l'entreprise pendant les prochaines années. En contrepartie, ce n'est pas un logiciel ou un site maintenable qui est demandé, mais il faut qu'il soit assez bien réalisé et fonctionnel pour être livré le plus rapidement possible afin d'en dégager une marge. Les éditeurs de logiciels quant à eux le louent généralement en tant que service ou le vendent en tant que bien. Le nerf de la guerre réside alors dans la maintenabilité du logiciel auquel des fonctionnalités doivent être ajoutées au fil du temps. Plus le logiciel grossit, plus il est compliqué de maîtriser les effets de bord ainsi que de comprendre l'intégralité du logiciel et de son historique. Ils ont cependant la chance d'avoir conçu une sorte de distributeur automatique qui n'a pas besoin d'action humaine pour fonctionner. Pour conclure, quel que soit le modèle d'entreprise, le développeur n'est que rarement en contact direct avec le client. Les chefs de projets sont en première ligne pour échanger avec eux. Les UX/UI peuvent faire intermédiaires en consultant le client, mais ils doivent en informer les chefs de projets.

Personne	Réalisation	Approbateurs	Consultés	Informés
Client	Besoin	Chef de projet	UX/UI	Développeur
Chef de projet	Cahier des charges fonctionnel et planning	Client	UX/UI	Développeur
UX/UI	Spécifications visuelles et ergonomiques	Client	Développeur	Chef de projet
Développeur	Code source	Chef de projet	Chef de projet	Chef de projet

F6 : Matrice « RACI » simplifiée, d'après mes 5 ans d'expérience dans l'industrie informatique

Il convient de planifier et prendre son temps. Les deux vont de pair, il ne sert à rien d'être dans la précipitation, car elle mène à l'erreur. Un travailleur peut être rapide, mais il ne lui faut pas oublier la rigueur, ou tenter de faire plusieurs choses à la fois. Il vaut mieux prendre une pause, que d'être happé par la réalisation d'une autre tâche et oublier de finaliser celle initiale. Et si je dois changer de tâche, alors il convient de la noter et mieux, la (re)planifier. Malgré la planification il faut aussi accepter que tout et n'importe quoi peut arriver à n'importe quel moment de la vie de chacun et donc savoir être flexible. C'est pourquoi le partage de connaissance est important en entreprise. Lors de mon alternance chez SoeMan j'ai pu participer à des réunions d'on-boarding après quoi nous réalisons des comptes rendus. À De Bussac Multimedia, nous pouvions initialement partager des articles de blogs, et plus tard présenter des diaporamas et démonstrations de sujets techniques dans des sessions afterwork. Actuellement chez ABGX, nous avons même tous nos vendredis après-midi dédiés à un projet interne et annexe de notre choix. Pour autant nous ne planifions que les tâches à faire, pas exactement quand. Nous définissons un ordre, une pile des tâches à réaliser, avec des niveaux de priorité et de complexité. Tous les workflows de travail que j'ai pu voir ont leurs avantages

et leurs inconvénients, pour autant c'est chez ABGX que j'ai personnellement retrouvé ce qui me convient le plus parmi toutes mes expériences. J'ai immensément apprécié être à plein temps sur la refonte graphique de SoEMan, mais je n'avais plus l'impression d'être utile dans l'avancement du logiciel durant cette période. Chez De Bussac au contraire, j'avais la sensation d'être en symbiose avec mon emploi du temps qui régissait mon activité professionnelle avec malgré tout un certain confort. Ce même confort m'a appris à prendre mon temps lors de mon travail à ABGX, m'apportant cependant une sensation de décélération qui m'a parfois frustré. Pour autant, c'est celle qui m'a remis en phase avec le travail. Dans le travail comme en voiture, il faut adapter sa vitesse à son environnement. Même sur une piste de course comme en formule 1, les meilleurs temps sont réalisés en ayant une vitesse et une trajectoire optimale, selon les conditions matérielles au moment donné. Enfin, j'ai souvent entendu dire « ce n'est pas ce qui est demandé ni prioritaire, on fera ça plus tard », et bien évidemment, la plupart du temps cela finit par n'être jamais fait. Dans les projets informatiques, c'est ainsi que la dette technique s'immisce, les fonctionnalités s'accumulent, les développements s'enchaînent, et l'équipe technique ne pallie aux problèmes qu'avec des corrections rapides, s'attaquant aux conséquences plutôt qu'aux causes. Dès que je constate un problème, j'en cherche la source, je le documente pour que l'information ne se perde pas, et ainsi facilite sa correction.

La réalisation d'un projet est donc un travail de groupe, mais aussi individuel, c'est pourquoi la formation du personnel est importante et que cette dernière passe par l'apprentissage personnel. Il convient, quel que soit votre métier, d'effectuer une veille stratégique pour vous-même avoir une démarche d'amélioration continue, et ce en dehors de toute formation organisée par votre établissement. Être autodidacte vous permet d'être maître de vos connaissances et de votre travail. Cela ne signifie pas pour autant qu'il faut tout faire seul et soi-même, car il y a généralement besoin d'autrui pour apprendre, comme pour tester ou mettre pleinement en application ses compétences.

Apprentissage et développement personnel

L'apprentissage commence généralement avec de la mémorisation, s'ensuit une compréhension qui permet la résolution de problèmes, et enfin s'y ajoute la créativité que nous débloquons lorsque nous avons le savoir cumulé au savoir-faire. La mémorisation étant le terreau de ce savoir, il est important de comprendre que les scientifiques ont aujourd'hui prouvé que les humains ont, tout comme les machines, une mémoire de travail et une mémoire à long terme. Ces deux mémoires sont comparables à la RAM et au stockage de masse. Le cerveau possède même un mécanisme d'encodage de l'information vers la mémoire à long terme et de récupération vers la mémoire de travail. Les informations les plus importantes possèdent plus d'amorces, permettant de les retrouver et s'en souvenir plus facilement, avec bien sûr certaines informations dont le signal se perd dans l'oubli, nécessaire à trier et jeter ou archiver ce qui n'est pas jugé important. Dans le cas de la mémoire à long terme, cet oubli se produit en perdant des amorces au fur et à mesure du temps. À l'aide de schémas mentaux, nous associons

un ensemble de concepts ou d'idées liées, permettant de formuler facilement des phrases compréhensibles et transmissibles. L'humain a la chance d'être multimodal et d'avoir des « périphériques » biologiques variés. Une fois qu'il a capté une information grâce à un d'entre eux, il doit la revoir ou se la répéter à intervalles réguliers et assez court afin de la mémoriser. D'après ce que nous avons vu précédemment, cela fait sens. La répétition de la perception de cette information par l'esprit va lui permettre de créer plus d'amorces et constituer des schémas mentaux. La reformulation sous forme de questions à se poser pour tester ses connaissances, ou de réexplications claires faites à autrui, permet un apprentissage plus rapide et diversifié. ³⁴

Bien que les compétences numériques, sociales et civiques, ainsi qu'apprendre à apprendre, fassent partie du socle commun des connaissances et des compétences de l'OCDE, je pense qu'elles sont négligées et que l'éducation nationale devrait enseigner en partant de choses plus concrètes comme le suggère John Dewey dans *School and Society*. J'ai dû attendre l'enseignement supérieur pour avoir un unique cours de quelques heures sur la mémorisation et l'autoformation. Actuellement, l'éducation n'apprend pas aux gens à bien vivre dans leur environnement. Imposer un programme est nécessaire pour avoir une base commune, mais les gens ont une vie à côté qui les impacte positivement ou négativement dans le scolaire. Certaines compétences qui devraient être universelles, comme la gestion d'un budget, d'un planning, la cuisine, l'entretien d'un foyer, ainsi que les démarches administratives élémentaires comme les déclarations d'impôts ou les demandes d'aides, ne sont transmises que par les parents ou les enseignements supérieurs spécialisés. Or selon moi le rôle de l'éducation devrait être de faire en sorte que toute personne ayant un objectif soit capable d'elle-même trouver les prérequis et les appliquer. De plus nous avons de formidables références nationales comme legifrance.gouv.fr, service-public.fr et gallica.bnf.fr, ou internationales comme fr.wikipedia.org et elles ne sont que trop peu consultées. Nul n'est censé ignorer la loi, mais seuls les professionnels la lisent et la connaissent. L'éducation ne peut pas tout faire, mais elle devrait enseigner comment trouver les outils et informations dont l'élève aura besoin, en plus de lui apprendre à apprendre, et ainsi comment s'autoformer. L'école et l'éducation nationale n'ont pas le monopole du savoir. Elles permettent cependant d'avoir des bases testées et certifiées, dans un périmètre défini par le programme qu'elles proposent.

Pour brièvement parler de la didactique, elle s'intéresse plus aux contenus même des cours, contrairement à la pédagogie qui représente la manière de présenter ses dits cours pour les transmettre d'un mentor à un apprenant. Les deux sont donc complémentaires, je pense que la didactique et son architecture sont fondamentales. Elles doivent être différentes en fonction de niveau et de la personnalité ou de l'état d'esprit de chaque élève, d'où l'intervention de la pédagogie pour diriger les étudiants vers des cours et exercices adaptés à leurs raisonnements, leur permettant ainsi d'en comprendre les concepts.

L'apprentissage se fait souvent par l'erreur, sans elle rien ne serait possible, car le succès ne s'obtient qu'en résolvant des microproblèmes. Malgré le fait que l'erreur soit bénéfique, des études montrent désormais que l'apprentissage se fait mieux par la récompense que la punition.³² Il est plus productif d'encourager un bon comportement que d'en réprimander un mauvais. Bien que cela soit parfois nécessaire, il vaut toujours mieux orienter sa pédagogie de manière positive. Je pense que la pédagogie active et mutuelle des pédagogues contemporains est une belle utopie inspirée de Rousseau. En réalité un bon apprentissage est avant tout du ressort de l'apprenant. C'est d'ailleurs pourquoi il est au cœur de ces techniques d'apprentissages. Depuis Montaigne, l'adage « *L'enfant n'est pas un vase que l'on remplit, mais un feu que l'on allume* » est couramment repris. En effet, la plus forte source de motivation est intrinsèque, c'est celle qui émane de la pensée telle la conscience. Celle qui est infinie comme l'imagination. Celle dont naît une volonté et une satiété de savoir intarissable. La pédagogie active et plus généralement l'autodidaxie permettent à l'élève d'être acteur de son apprentissage. Il choisit ce qu'il désire apprendre, quand, et comment ; bien que cela se passe souvent par le jeu dirigé puis libre. La motivation de l'étudiant devient intrinsèque au fur et à mesure qu'il acquiert de la liberté, ce qui favorise l'autonomie, la retenue, la prise de décision, et la gestion des conflits. L'adulte ou le mentor ne doit être qu'un médiateur entre l'élève et le savoir, un guide d'apprentissage. Il doit selon moi lui faire part de ce qu'il sait vrai, le démontrer, et admettre qu'il peut avoir tort. Le vrai respect ne s'obtient pas par l'autorité, mais par le partage, tout en respectant la barrière sensible nécessaire à toute relation professionnelle.

Les logiciels éducatifs sont également un médium guidant l'apprentissage. Comme j'ai pu le dire précédemment, j'ai appris à lire à 2 ans et demi et sauté deux classes grâce à « Reader Rabbit, Learn to read with phonics »³⁵. Je n'en suis pas un génie pour autant, et ce n'est pas pour faire preuve d'humilité. J'ai juste expérimenté un programme, réalisé par une équipe logicielle qui avait une conscience de l'expérience utilisateur, et qui s'est mise à la place de leurs joueurs. Ils ont alors inventé un scénario où un ordinateur propose un vœu à une petite souris, qui ne sachant ni appréciant lire, lui demande alors de ne plus jamais avoir à le faire. L'ordinateur retire alors l'intégralité des caractères écrits de sa ville, et le joueur doit retrouver les lettres de l'alphabet pour écouter le son qu'elles produisent, ainsi que les mots qu'elles permettent de former. Ce procédé guide l'élève pour qu'il construise au fur et à mesure, à travers le jeu vidéo, des associations entre les caractères visuels, les sons, et les formes qu'elles représentent dans le sens d'idées. Cela développe ainsi la lecture textuelle et la compréhension qui en découle. De manière générale, le jeu accroît les compétences sociales et cognitives de l'apprenti, qui n'est plus passif face à un cours qui lui est dispensé par une autorité qui sanctionne et récompense, mais découvre et s'approprie la connaissance par lui-même. Bien que le jeu libre ait des avantages, le jeu dirigé permet quant à lui de structurer l'apprentissage à l'aide d'un mentor favorisant l'acquisition de compétences qu'il transmet à ses disciples. Ce cadre réside dans l'analyse des difficultés et problèmes de l'étudiant, par la recherche des notions manquantes pour y pallier, et l'élaboration d'un plan permettant d'y arriver, composé de cours, d'exercices, et enfin d'un examen, nécessaire à la validation individuelle de la théorie.

Cette notion d'apprentissage actif se retrouve dans les tutoriels de langages de programmation comme avec *Rust par l'exemple* ou *Rustlings*, mais aussi dans des logiciels en ligne utilisant le jeu pour faciliter l'apprentissage. Je pense à Scratch, FlexboxFroggy, et CodingGame.

La tablette pourrait remplacer les manuels scolaires, les cahiers, réduire les coûts et le poids des cartables. Bien que certains pays comme la Suède l'aient essayé avant de marche arrière, je pense qu'avec un ludiciel adapté couvrant l'ensemble des matières de tout niveau à l'aide d'un arbre de compétence, il serait possible d'harmoniser les enseignements et d'appliquer les programmes tout en réduisant le temps d'apprentissage. Quand j'ai appris à écrire, les périphériques tactiles n'étaient pas communs, j'avais alors un retard sur cette compétence. Il existe aujourd'hui une multitude d'applications mobile pour apprendre à écrire. Sur une tablette possédant un stylet, un enfant peut jouer en toute autonomie pour acquérir cette habileté. Le problème et la crainte des enseignants comme des parents, sont essentiellement la casse, conjointe au prix du renouvellement d'un tel appareil électronique, accompagné des dérives liées aux possibilités illimitées d'un ordinateur, pouvant distraire l'élève et le détourner de ses objectifs éducatifs. Je leur répondrais que leur rôle est justement d'enseigner la précaution et la discipline, l'élève pourra ainsi se charger lui-même du reste. Enfin je terminerai cette partie sur l'apprentissage avec une citation de jeu vidéo. Dans LoL, Maître Yi dit « *Un vrai maître est un éternel étudiant* ». Je rajouterai aussi un conseil personnel que j'applique depuis toujours : « *N'apprenez pas par cœur, comprenez et retenez la logique qui produit le bon résultat* ».

Pour continuer sur le développement personnel, j'ai très tôt appris à cuisiner, et je continue aujourd'hui de pratiquer cette discipline afin d'améliorer ma propre expérience culinaire ainsi que mes connaissances annexes. L'adage que j'ai introduit à la fin du paragraphe précédent s'applique parfaitement à la cuisine, dont les recettes sont vouées à ne pas être suivies à la lettre. Nous pouvons les modifier et les adapter en fonction des goûts ou ingrédients de chacun, à condition d'avoir une observation attentive et une compréhension pluridisciplinaire. La gastronomie est un art, une philosophie, et une science combinatoire. C'est un grand tout. Depuis l'antiquité les recettes sont des formulations logiques, les ingrédients sont biologiques, leurs modifications par leurs préparations, leurs mélanges, et leurs méthodes de cuisson sont physiques ou chimiques. Enfin les proportions sont quant à elles mathématiques.

Nous savons aujourd'hui scientifiquement que notre alimentation conditionne notre santé, et inversement. Tout être vivant a besoin d'une entrée d'énergie constante pour contrebalancer son activité. Sans cela ou avec une mauvaise alimentation, il paraît logique d'avoir un système défaillant. C'est la source même de la vie. La cuisine a quelque chose d'universel, tout peuple de la Terre a inventé indépendamment des autres ses propres recettes, cultures, élevages, croisements d'espèces volontaires ou non. La mondialisation est venue optimiser l'agriculture et permettre des échanges de nourriture et de pratiques culinaires à l'échelle de la Terre. Les scientifiques ont découvert que nous vivons en symbiose avec des bactéries qui nous aident à digérer les aliments et qu'elles représentent une partie non négligeable de notre poids.

En informatique, et notamment en intelligence artificielle, les résultats dépendent aussi des informations qui alimentent le système. Le résultat d'un algorithme ou d'une fonction mathématique dépend des variables passées en entrée. Le développement personnel est également celui d'une pensée que nous devons alimenter. Mais à quoi cela sert de penser ? Pour y répondre, je vais citer un logicien introduit précédemment. Il déclare que l'action même de penser est stimulée par l'irritation du doute, et ne s'arrête qu'en achevant son raisonnement avec la production d'une croyance, qui est ainsi, la principale fonction de la pensée.

« It was there noticed that the action of thought is excited by the irritation of doubt, and ceases when belief is attained; so that the production of belief is the sole function of thought. »

Charles S. Peirce — *How to make Ideas clear*

Nos croyances ont une influence sur nos vies. J'ai longtemps été contre l'idée d'une pensée positive, et je la vois parfois comme une tentative de s'autodétourner cognitivement (gaslight). Pour autant, il a été scientifiquement prouvé que les croyances positives ont une influence positive, et que les croyances négatives ont une influence négative. Cela correspond respectivement aux effets Pygmalion et Golem. Grâce à l'imagination, nous envisageons différents scénarios possibles. Faire preuve d'un réalisme optimiste, envisageant le mal comme le bien afin d'éviter les problèmes et d'améliorer ce que peut l'être, permet **l'anticipation**. Pour résumer, selon moi la question fondamentale à se poser face à une information douteuse serait « Est-ce scientifiquement démontrable ? ». Celle à se poser vis-à-vis d'une croyance serait quant à elle « Que se passerait-il si je me mettais à croire ça ? ». Enfin, si une éventualité devient assez probable, je me demande alors : « Si cela arrive, que se passera-t-il ? »

Dans le cadre du logiciel, l'anticipation se produit notamment avec la compilation qui nous préserve de la plupart des erreurs d'exécution. Elle peut également se manifester grâce à la gestion de projet, mais surtout en travaillant en binôme comme le recommande *l'Extreme Programming* de Kent Beck. Coder en binôme permet de tester ses idées face à autrui de manière dialectique, c'est-à-dire communiquer en tête à tête de manière à partager des connaissances et du savoir-faire. Cela doit idéalement se faire de manière claire, honnête, consistante, réactive, et adaptée à l'interlocuteur. En communication comme en électronique, un ensemble de signaux faibles peuvent s'entre-interférer et créer du bruit. Un signal fort peut quant à lui perturber voire totalement stopper un système, et c'est parfois l'effet escompté. Tout étudiant l'a déjà expérimenté, nous avons tous déjà été dans une situation où un groupe bruyant se tait après qu'une autorité, généralement le professeur, ait haussé la voix ou sifflé. Si un professeur y a recours, c'est que les bavardages sont des informations parasites à son cours. De la même manière que les élèves devraient être attentifs, les salariés d'une entreprise devraient tous être concernés par ce qu'ils produisent. Norbert Elias a déclaré que l'individu et la société ne sont pas deux choses séparées, et je pense que nous sommes tous responsables de nos actes quels qu'ils soient, y compris l'acte de rester passif et ne rien faire ou dire.

Metagame

Lorsque des joueurs poussent assez l'analyse d'un jeu, ils arrivent à en comprendre assez bien le fonctionnement pour être capables d'utiliser les règles pour en profiter. Ils sont à même d'en abstraire les principes, de comprendre la logique sous-jacente, et peuvent alors avoir une approche stratégique leur permettant de dépasser les limitations fixées ou espérées par les concepteurs du jeu. L'idée du métajeu est d'appliquer la science et la théorie des jeux de manière à trouver une technique dont les chances de victoires sont les plus hautes possibles. Même sur les jeux les plus simples comme Tetris, les joueurs ont trouvé des techniques de frappe comme *l'hypertapping* ou le *rolling*, permettant d'effectuer plus d'actions par secondes avec leurs manettes. Si bien qu'ils ont ainsi pu aller au-delà du niveau « maximum » à partir duquel la vitesse de chute des formes s'arrête d'accélérer.

Toute forme de vie répond à une logique très simple. Comment maximiser mes récompenses en minimisant mes efforts ? Comment obtenir plus de plaisir et moins de souffrance ? La vie ne recherche que plus de plus et moins de moins. En tant que futur chef d'entreprise, comment puis-je, à l'aide de l'intelligence économique ou d'autres disciplines comme la théorie des jeux, analyser les situations qui se présentent à moi pour effectuer les meilleurs choix ? Il faut s'informer, se protéger des potentielles agressions, et influencer son environnement en sa faveur. Les questions à se poser rejoignent les techniques présentées dans le chapitre « Décisions et gestion de projet », elles sont très simple et au nombre de quatre : De quoi s'agit-il ? Quels sont les enjeux ? Qu'est-ce que je peux faire ? Qu'est-ce que je dois faire ?

Dans le cas des investissements financiers et jeux d'argents, la formule de Kelly peut permettre une bonne approximation de la mise à investir, généralement celle-ci avoisine le tiers du capital total. Elle permet à un joueur d'éviter un excès d'enthousiasme ou de pessimisme et ainsi de raisonner scientifiquement son jeu pour l'équilibrer. Malgré tout le biais peut aider, car si j'ai été biaisé alors d'autres ont pu l'être aussi et cela peut contribuer à l'intérêt donné à une valeur financière, définie par l'offre et la demande. Mais les formules sont paradoxalement trop mathématiques pour représenter précisément le monde humain multifactoriel et imparfait. Nous ne pouvons qu'approximer la réalité. Comme décrit en fin de section précédente, la croyance à tout de même un grand pouvoir, les individus ont donc tout intérêt à croire en eux et leurs proches, ainsi qu'en leurs choix. Il faut cependant le faire rationnellement, en testant ses croyances, et si elles en deviennent négatives, il convient de les changer, voire tester l'inverse. Toujours en restant tempéré et raisonné. Il n'y a aucune formule miracle, il faut juste tester et voir ce qui marche et ce qui ne marche pas, tout en gardant une approche logique itérative.

La vie est un jeu, il est donc possible d'y appliquer les mêmes notions, alors jouez, et s'il le faut, suivez le pacte d'Ulysse. La métacognition et la rétroaction permettent de se comprendre soi-même et d'adapter ses comportements et habitudes. Il faut perpétuellement s'efforcer d'être le plus conscient possible de sa situation, des possibilités et opportunités qui se présentent à nous.

Bilan actuel

En décidant de reprendre mes études après 2 ans et demi de CDI, j'ai décidé d'investir une partie de mon temps dans l'acquisition des informations nécessaires à la réalisation de mon objectif professionnel futur. À cet effet, je me suis renseigné sur ce qui était à l'origine même de la programmation, l'informatique et plus globalement de la gestion de l'information. Avant de répondre pleinement à la problématique, je vais donc faire un bilan de l'informatique à notre époque. Comme indiqué en introduction de ce chapitre, elle a été popularisée jusqu'à être désormais omniprésente. La terre est entourée de satellites en orbites, les continents sont interconnectés avec des câbles sous-marins, 7 personnes sur 10 possèdent un smartphone, les demandes d'informations et démarches administratives se numérisent rapidement. La programmation est facilitée par l'apparition de langages de haut niveau à la syntaxe simplifiée. Depuis 1950, bien que les interfaces graphiques aient permis de faciliter l'utilisation des ordinateurs, la programmation n'a plus été révolutionnée depuis les premiers langages de haut niveau. Seule l'apparition du langage Rust, dont l'adoption dans les entreprises sera encore longue, a permis d'avoir un nouveau candidat pour ce poste. Il a des performances équivalentes au C et une expérience de développement plus agréable que les autres, et ce grâce à ses nouveaux concepts de gestion de la mémoire (évitant ainsi 70% des bugs selon Microsoft³⁶), et un compilateur plus explicite. Pour le reste, la majorité des langages ont leurs propres qualités, notamment grâce à leurs communautés et aux outils qu'elles ont développés. Pour exemple, Python est toujours une référence pour le traitement de la donnée, et JavaScript a standardisé la gestion des interactions utilisateurs et la dynamisation des interfaces graphiques.

Le matériel quant à lui a beaucoup évolué. Les ordinateurs se sont petit à petit miniaturisés, devenant portables et mobiles, offrant même désormais des expériences immersives réalistes grâce aux casques de réalité virtuelle. Depuis les années 2000, l'automatisation et la robotique s'immiscent dans notre quotidien et dans les ménages à travers la domotique. Robots aspirateurs, drones munis de caméras, voitures autonomes... Nos outils de tous les jours deviennent indépendants et réduisent notre charge de travail. Mais ce qui nous intéresse dans le cadre de la problématique, ce n'est finalement pas tant le matériel, mais l'information et la logique qui en émanent. La population mondiale utilise de plus en plus les technologies de l'information et internet. C'est pourquoi la grande majorité des entreprises se modernisent et veulent avoir une gestion informationnelle fiable accompagnée d'une visibilité mondiale fournie par les réseaux informatiques. Les recherches et compréhensions récentes dans le domaine de la bio-informatique ou de la physique quantique ont permis, sur les idées respectives de Richard Feynman et Rolf Landauer, de mettre en œuvre de nouvelles manières de stocker de l'information que ce soit dans de l'ADN³⁷, ou dans des bits quantiques appelés qubits. La théorie des nœuds dont j'aurais aimé plus parler a trouvé des applications dans l'étude des protéines et plus généralement des molécules. Grâce à ces études, les scientifiques ont été capables d'inventer de nouvelles structures moléculaires plus solides que le kevlar. Ces nouvelles technologies ouvrent de nouveaux horizons, repoussant les limites du possible en

solutionnant de multiples problèmes, nous permettant par la suite d'en trouver de nouveaux. Les systèmes de communication sont rapidement passés du télégraphe à l'appel téléphonique, jusqu'à l'avènement d'internet à la fin du siècle dernier. Cette mondialisation de l'information et son instantanéité nous permettent aujourd'hui de collaborer n'importe quand avec n'importe qui, où que nous soyons, d'avoir accès à des connaissances et de trouver réponses à nos questions.

Les découvertes ne sont pas toujours l'œuvre de savants ou de personnes renommés. En 2023, de simples utilisateurs de jeux mobiles découvriront ce que les mathématiciens recherchent depuis des siècles. Ça a été le cas du problème einstein essayant de trouver une forme géométrique unique dite d'*ein stein* (une tuile en allemand), que les chercheurs ne pensaient pas possible jusque là. En réalité elle est très simple et peut se construire en découpant trois hexagones liés en suivant les médiatrices de leurs côtés. Cette dernière ressemble à un t-shirt ou un chapeau et permet de réaliser un pavage aperiodique à tuile unique c'est-à-dire de recouvrir une surface plane par un emboîtement de copies de cette forme, sans que son motif ne se répète à l'identique comme les carreaux d'une grille. Pour ce faire il faut parfois avoir recours à une tuile inversée, problème qui a été réglé avec la découverte d'une autre forme, le spectre.

Enfin l'humain est de plus en plus capable de comprendre ou restituer des faits **à postériori**. Nous sommes même en mesure de restaurer des informations perdues. Les technologies comme les rayons X et le Lidar sont très utilisés en archéologie afin de ne pas avoir à abîmer des œuvres scellées. Plus récemment, il s'est avéré que les techniques de peinture des grands maîtres ont été transmises par Léonard de Vinci qui appliquait déjà une première couche à base de plomb nommée Plombonacrite³⁸. Enfin, les intelligences artificielles ont permis de déchiffrer des parties d'un manuscrit calciné lors du vesuvius challenge³⁹.

En conclusion, l'informatique moderne, comme toutes les technologies, a d'abord exploré les possibilités en créant de multiples outils et techniques, puis sa démocratisation a nécessité une standardisation. Des consensus se sont alors formés, ils ont établi des normes, simplifié et harmonisé les outils et techniques en les rendant universels. C'est le cas du web avec le W3C qui a standardisé HTML, CSS, JS, ou des protocoles réseau comme TCP/IP, HTTP, et j'en passe. Tous établissent des bases communes à suivre pour établir des systèmes de communications sécurisés. C'est pourquoi je vais dès maintenant présenter les techniques de développement qui sont actuellement en place pour la réalisation de projets logiciels web.

Concepts et outils de programmation

Imaginons que vous vouliez réaliser un site web et qu'il soit accessible en ligne. En réalité il vous suffit d'une connexion internet, d'un logiciel de serveur, et enfin d'un nom de domaine pour y accéder sans avoir à mémoriser une adresse IP. C'est la base de l'application web proposée en solution à la problématique. Je présenterai cette dernière à la suite de cette section dans laquelle je vais tâcher d'expliquer son processus de création le plus simplement possible.

Réalisation et mise en ligne d'un site web

Pour un utilisateur averti, la démarche peut ne prendre que quelques minutes. Après toute cette rédaction textuelle, une petite démonstration technique s'impose. La première étape consiste à réaliser un site web. Il suffit pour cela de trois composantes réparties dans trois fichiers, ou centralisées en un seul dans le cas de ce prototype. Cette étape est donc logiquement séparée en trois étapes, qui correspondent aux trois parties d'un site web, le contenu structuré, l'habillage, et la réaction que doit avoir le site face aux entrées de l'utilisateur. Pour l'exemple je vais réaliser un site très simple, un mot, centré au milieu de l'écran, qui inverse la couleur de fond et celle du texte lorsque je clique sur ce dernier.

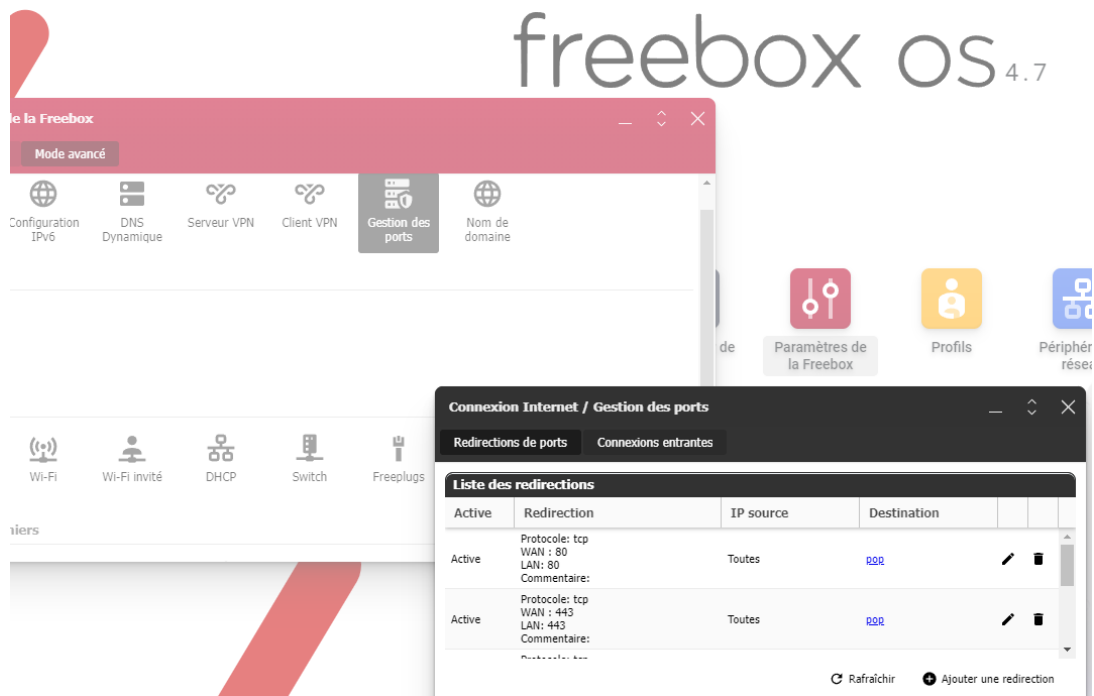
```
<html>
  <!-- Ceci est un commentaire HTML qui ne sera pas interprété par le navigateur -->
  <!-- Le texte à l'intérieur de la "tête" du site est interprété mais sera pas affiché -->
  <head>
    <!-- Récupération de la fonte https://fonts.google.com/specimen/Alice -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Alice&display=swap"
rel="stylesheet">

    <!-- Script d'habillage CSS -->
    <style>
      * { box-sizing: border-box; transition: all 0.5s ease-in-out; }

      body {
        height: 100vh; background-color: #FEFEFE;
        display: flex; justify-content: center;
        align-items: center;
      }
      body.inverted { background-color: #333; }

      h1 {
        font-family: 'Alice', serif;   cursor: pointer;
        font-size: min(15vw, 75vh);   color: #333;
        transition: all 0.5s ease-in-out;
      }
      .inverted h1 { color: #FEFEFE; }
    </style>
    <script>
      function invert() {
        document.body.classList.toggle("inverted")
      }
    </script>
    <!-- Notez que j'aurais pu importer le style css et le script js depuis un fichier -->
    <!-- grâce à une balise link ayant un attribut href="chemin/vers/fichier.ext" -->
  </head>
  <!-- Le corps du site décrit la structure de ce qui s'affichera à l'écran. -->
  <!-- Dans notre cas, un simple titre de niveau 1 cliquable -->
  <body>
    <h1 id="text" onclick="invert()">Bonjour</h1>
  </body>
</html>
```


La deuxième étape est d'ouvrir les ports. Allez dans le panneau de configuration de votre box internet et ouvrez les ports qui permettent d'émuler un site web et d'écouter les requêtes des clients en HTTPS avec le numéro 443 et en HTTP avec le numéro complémentaire 80.



F7 : Accès à la gestion des ports dans l'interface freebox OS 4.7

Pour l'étape trois, je vais configurer un nom de domaine. Cela permet de relier une [adresse IP](#) à un alias textuel plus simple. Pour arriver sur un site internet, nous ne retenons jamais son adresse IP, mais son nom de domaine comme `google.fr`.

Pour faire cette liaison, le plus simple serait que votre box internet possède une IP fixe, ou une option pour qu'elle le soit. Si c'est le cas alors il suffit d'aller sur le site où vous avez réservé votre nom de domaine et ajouter une entrée DNS de type A si vous avez une adresse IPv4, ou AAAA si votre adresse IP est en version 6.

4 Enregistrements DNS Ajouter un enregistrement					
Service	Source	Type	Cible	TTL	Dernière mise à jour
Serveur DNS	editide.fr	NS	ns31.infomaniak.com	1 h.	08/01/2020 13:43:24
Serveur DNS	editide.fr	NS	ns32.infomaniak.com	1 h.	08/01/2020 13:43:24
Adresse web	editide.fr	A	82.65.249.100	1 h.	17/12/2022 14:42:33
Adresse web	dev.editide.fr	A	82.65.249.100	1 h.	18/12/2022 11:46:04

F8 : Interface de configuration DNS d'Infomaniak

Si votre adresse IP est susceptible de changer, les hébergeurs proposent généralement des solutions de DNS dynamique. Le concept est simple, lorsque votre adresse IP change, votre serveur doit contacter le service DNS et lui communiquer sa nouvelle adresse IP. Avant de pouvoir obtenir une adresse IP statique par le biais de mon fournisseur d'accès internet, j'ai eu recours à ce genre de méthode en utilisant l'API d'Infomaniak. Vous pouvez retrouver mon code sur : <https://github.com/hedocode/nodeInfomaniakDynamicDNS>

La quatrième et dernière étape consiste à mettre en place un serveur HTTPS. Pour cela j'ai téléchargé **NGINX**, puis Certbot en suivant les [instructions](#). Une fois installé, je m'assure que mon serveur NGINX est à l'arrêt dans le gestionnaire des tâches (**CTRL + Shift + échap**) et je lance un terminal pour créer un certificat avec la commande **certbot certonly --standalone** . Il ne me reste alors plus qu'à importer le certificat et la clé privée dans un bloc **server** de ma configuration NGINX puis je définis l'emplacement de ma page web à l'aide de la directive **root** et **location** .

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    keepalive_timeout 65;
    server_names_hash_bucket_size 64;

    server {
        listen 443 ssl;
        server_name bonjour.blanchardorian.fr;

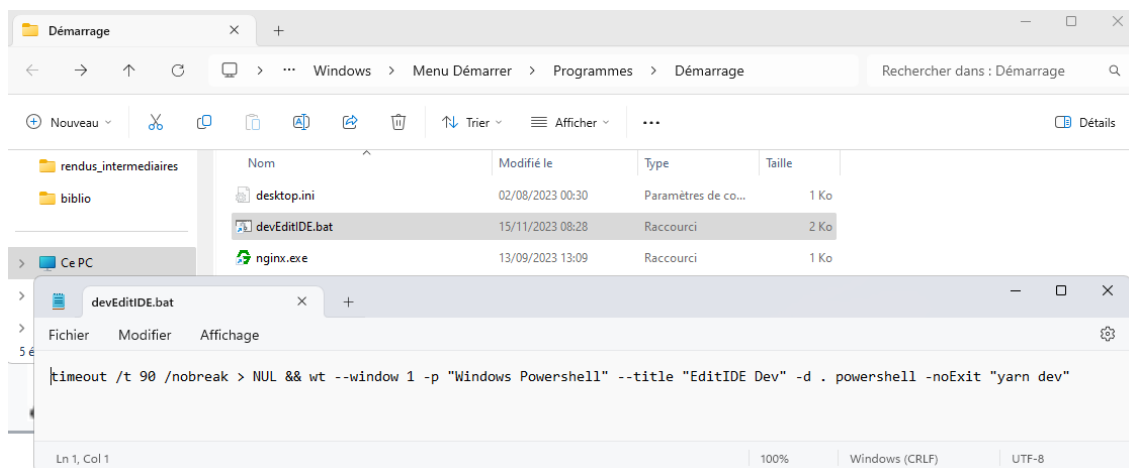
        ssl_certificate C:/Certbot/live/bonjour.blanchardorian.fr/fullchain.pem;
        ssl_certificate_key C:/Certbot/live/bonjour.blanchardorian.fr/privkey.pem;

        root C:/Users/admin/Desktop/bonjour;

        location / {
            index index.html;
        }
    }
}
```

L'utilisation de NGINX comme serveur me permet de l'utiliser comme reverse-proxy, c'est-à-dire que je peux rajouter autant de serveurs que je veux sur une même machine et y accéder avec des noms de domaines différents. Pour cela il me suffit de copier-coller un bloc **server** et de le configurer en fonction de mes besoins.

Pour finir cette démonstration, je vais vous présenter deux autres mécanismes que j'utilise. Premièrement, le **lancement du site au démarrage** de mon serveur. Pour un Linux, j'aurais fait un service ou daemon. Sur Windows, il suffit de trouver le dossier Démarrage ou Startup. Il est situé dans le dossier utilisateur, quelque part dans le fameux `AppData\Roaming`. La solution la plus facile pour y accéder est de faire `Windows + R`, de taper `shell:startup`, et de valider. Vous pouvez ensuite épingler le dossier au menu d'accès rapide pour le retrouver plus facilement. Tous les raccourcis que vous mettez dedans seront exécutés au démarrage. Dans mon cas je lance automatiquement NGINX et attends 90 secondes avant de lancer le script de démarrage de mon serveur NodeJS dans la fenêtre numéro 1.



F10 : Contenu d'un script situé dans le dossier de démarrage Windows

La dernière astuce est celle que j'utilise le plus. Je suis l'heureux détenteur d'un ordinateur portable Asus Zenbook qui est très compact, mais a l'inconvénient de beaucoup chauffer. À mon arrivée à l'École Hexagone, j'ai donc décidé d'acheter un mini PC et l'utiliser à distance. À cet effet, j'ai très simplement activé le **bureau à distance** de la version Windows Professionnel directement depuis ses paramètres. Ensuite, il suffit d'activer le port 3389 comme indiqué plus tôt. Si vous le faites, il faut cependant avoir un mot de passe avec assez d'entropie. Pour cela, il doit être assez long et contenir assez de caractères différents, incluant des minuscules, majuscules, chiffres, et caractères spéciaux. Malgré cette disposition, des botnets peuvent tenter de réaliser de multiples tentatives de connexion ou trouver une faille de sécurité dans le protocole et interférer avec votre système. Il convient donc de ne pas ouvrir le port 3389 et d'installer un VPN pour accéder au réseau local de manière sécurisée. L'avantage du bureau à distance est que je peux à tout moment retrouver ma session telle que je l'avais laissée, sans affecter le travail qui est en cours dessus, et ce depuis différents appareils.

Langage logique

Nous parlons communément un *langage naturel* et spontané. Bien qu'il possède des règles de conjugaison, et de grammaire qui englobe la syntaxe commune à tous les langages ; le langage parlé ne nécessite pas pour autant de suivre ses dites règles à la lettre pour être capable de communiquer entre humains, car nous sommes capables d'interpréter et de traduire si besoin. À contrario, le *langage formel* ou construit, comme le code informatique, nécessite quant à lui une rigueur stricte, sans quoi il ne peut être interprété par l'interlocuteur qui est une machine ou un autre développeur. Que la langue soit usuelle ou permette de communiquer des instructions logiques à une machine, il est nécessaire d'en apprendre et respecter suffisamment la syntaxe, ses règles et concepts, pour s'en servir de manière à être compris. Situé entre le langage naturel et le code, le Lojban est une langue construite qui peut être parlée et permettrait même de communiquer de manière logique avec une machine.

L'interface la plus simple informatiquement parlant est la ligne de commande. Elle a un avantage, celui d'être totalement textuelle. Elle permet de dialoguer directement avec la machine en utilisant le clavier et les mots clés qu'elle comprend, c'est-à-dire les commandes qu'elle propose. Les langages de programmation permettant de coder comprennent quant à eux des données aux types communs. Comme évoqué précédemment, les deux grandes catégories d'informations sont quantitatives et qualitatives. Parmi les informations, nous distinguons donc les chiffres des autres données. En JavaScript, que ce soit avec ou sans virgules, ils correspondent tous deux au type `number`. Les informations qualitatives peuvent être des mots, composés de caractères, c'est-à-dire des fils ou chaînes de caractères mis les uns à la suite des autres. Les données textuelles seront donc ici des types `string`. La majorité des langages possèdent un type pour stocker des structures de données, la plus simple est l'`array`. C'est une liste pouvant contenir tous les types, y compris d'autres listes pour donner des tableaux ou matrices. Enfin, si nous voulons représenter une entité capable à la fois d'avoir des attributs nommés avec les types précédents et également des actions au travers des `function`, nous créerons un `object`. Les informations de telles structures imbriquées forment un **arbre** appelé **graphe** en mathématiques discrètes. Je vais expliquer cela plus en détail dès maintenant au travers de sections écrites en JavaScript (JS). Vous pouvez tester vous-même le code en faisant des copiés-collés dans la console d'un navigateur internet. Pour y accéder, il suffit d'être sur ordinateur et de taper sur la touche F12 ou faire la combinaison `Ctrl + Shift + I`. Si vous êtes sur mobile, vous pouvez facilement trouver un Sandbox ou Playground JavaScript en ligne.

Comme évoqué lors des [prémices de l'Algèbre](#), et plus généralement en Mathématiques, les informations sont stockées dans des *variables* (`var`), souvent représentées par une lettre, ou dans le cas de l'informatique, sous forme de mots. Parmi les variables, nous avons aussi une variante qui limite la visibilité à son contexte (`let`), et des *constantes* (`const`) qui ne changeront plus une fois qu'une information leur sera affectée.

Je vais donc commencer par créer une variable dans laquelle je vais stocker un objet représentant une voiture, dont les informations seront discrétisées à une marque, un modèle, une vitesse actuelle et maximum.

```
// Bienvenue dans du code ! Tous les textes d'une ligne situés après deux slash (//) sont des
// commentaires qui ne seront pas utilisés par le compilateur ou l'interpréteur qui permet à la
// machine de lire et convertir le code en langage machine pour enfin l'exécuter.

// Je peux ainsi l'utiliser pour communiquer avec tout lecteur de code,
// y compris le moi du futur qui aura oublié avoir écrit ça !

// Je vais donc déclarer une variable et lui affecter une valeur.
var ma_première_voiture = { // L'accolade ouvrante représente un objet.
  "marque": "Citroën",    // J'y déclare ses attributs, composé d'un nom,
  "modele": "C4",         // séparé d'une valeur par deux points ":",
  "vitesse_actuelle": 0,  // et que je sépare avec des virgules.
  "vitesse_max": 140
}; // Quand j'ai fini de déclarer l'objet, je referme l'accolade.
```

Les langages de haut niveau permettent de déclarer des *fonctions*. Comme en mathématiques et comme son nom l'indique, c'est en fonction d'une ou plusieurs variables fournies ou non en entrées, et de la logique qu'elle contient, que j'obtiens un résultat en sortie. Elles permettent généralement de réaliser des actions sur des objets. Dans notre cas, nous allons faire accélérer la voiture qui sera pour l'instant passée en paramètre.

```
// Les paramètres se mettent entre parenthèses.
function faire_accélérer(voiture) { // Cette accolade déclare l'ouverture d'un contexte.
  // Je peux récupérer des variables telles qu'elles s'appellent dans l'objet défini
  // précédemment, soit une avec un point, soit plusieurs avec des accolades.
  const vitesse_de_la_voiture = voiture.vitesse_actuelle;
  const { vitesse_max } = voiture;

  // Si la vitesse de la voiture ne dépasse pas la vitesse maximale après accélération.
  if(vitesse_de_la_voiture + 1 <= vitesse_max) {
    // J'attribue la valeur après accélération à la vitesse de la voiture.
    voiture.vitesse_actuelle = vitesse_de_la_voiture + 1;
  }
}

// J'affiche la vitesse actuelle de la variable `ma_première_voiture` qui contient son objet.
console.log("Vitesse initiale : %o", ma_première_voiture.vitesse_actuelle); // 0
// J'appelle la fonction en lui passant la voiture en paramètre pour qu'il la fasse accélérer.
faire_accélérer(ma_première_voiture);
console.log("Vitesse après accélération : %o", ma_première_voiture.vitesse_actuelle); // 1
// Bilan : La voiture a accéléré de 1 kilomètre !
```

Dans le cas présent, je suis en train de faire avancer la voiture, c'est un peu comme si je devais la pousser. Sachant cela, je peux vérifier que sa vitesse a augmenté, mais je vais surtout créer une *classe* pour la décrire de sorte qu'elle puisse se déplacer elle-même, et que je puisse facilement faire plein de voitures possédant les mêmes attributs. Vous noterez que dans le code comme dans les langages parlés, il y a une forme passive et active. La classe est comme un schéma ou un moule permettant de décrire un l'objet qui y sera créé avec l'affectation de ses attributs et de ses fonctions. En JavaScript la classe peut faire référence à l'objet qu'elle décrit à l'aide du mot clé `this`. Toute classe a une fonction obligatoire nommée `constructor`, elle indique l'algorithme à suivre pour créer l'objet qui résultera d'une nouvelle création avec le mot clé `new`. L'exemple ci-dessous va donc déclarer une classe Voiture que je peux construire en lui passant l'objet précédemment déclaré.

```
class Voiture {
  constructor(
    { marque, modele, vitesse_actuelle, vitesse_max }
  ) {
    // J'attribue les valeurs une à une.
    this.marque = marque;
    this.modele = modele;
    this.vitesse_actuelle = vitesse_actuelle;
    this.vitesse_max = vitesse_max;
  }

  accélère() {
    // Je peux renommer une variable de deux manières
    const vitesse_de_la_voiture = this.vitesse_actuelle;
    const { vitesse_max: vitesse_maximum } = this;

    if(vitesse_de_la_voiture + 1 <= vitesse_maximum) {
      this.vitesse_actuelle = vitesse_de_la_voiture + 1;
    }

    // À l'instar du constructeur qui le fait par défaut,
    // ici je retourne explicitement l'objet.
    return this;
  }
}

const object_premiere_voiture = new Voiture(ma_premiere_voiture).accélère;
console.log("Vitesse maintenant : %o", ma_premiere_voiture.vitesse_actuelle); // 2
```

La plupart des langages-objets ont un mot clé `interface` pour définir les méthodes communes à une famille d'objets. En JavaScript elle n'existe pas, il faudrait TypeScript pour cela. Dans mon cas je pourrais juste faire une classe abstraite avec un constructeur vide et donc sans attribut. Un objet créé avec cette classe n'aurait que des méthodes, car l'interface ne définit que des méthodes, leurs noms, paramètres et types de retour. Je ne vais donc pas utiliser d'interface, mais une abstraction d'héritage afin de définir un objet `Vehicule` qui se discrétise uniquement à la vitesse actuelle, maximale et sa gestion. Ainsi je pourrais réutiliser des fonctions qui ont un comportement par défaut ou configurable, et m'assurer que plusieurs éléments d'une même liste, dans notre cas des véhicules différant (voiture, bus, train, etc.), puissent tous effectuer le même calcul ou traitement donné. Par exemple sur la vitesse et la position pour les déplacer avant chaque affichage d'une simulation routière. Je peux ainsi redéfinir ma voiture plus simplement. De plus, en utilisant les abstractions, nos objets deviennent `super` !

```
// Si j'ai bien fait mon travail en amont, vous devriez être en mesure de lire ce code ! :)
class Vehicule {
  constructor(
    { vitesse_actuelle, vitesse_max }
  ) {
    this.vitesse_actuelle = vitesse_actuelle;
    this.vitesse_max = vitesse_max;
  }

  accélère() {
    const vitesse_de_la_voiture = this.vitesse_actuelle;
    const { vitesse_max } = this;
    if(vitesse_de_la_voiture + 1 <= vitesse_max) {
      this.vitesse_actuelle = vitesse_de_la_voiture + 1;
    }

    return this;
  }
}

class Voiture extends Vehicule {
  constructor(voiture) {
    super(voiture);

    const { marque, modele, vitesse_actuelle, vitesse_max } = voiture;
    this.marque = marque;
    this.modele = modele;
  }
}

const object_premiere_voiture_abstrait = new Voiture(ma_premiere_voiture).accélère;
console.log("Vitesse maintenant : %o", object_premiere_voiture_abstrait.vitesse_actuelle); // 3
```


Principes SOLID

La programmation-objet offre donc la possibilité de structurer les données, les fonctions et sa logique. Il est pourtant facile de l'utiliser incorrectement et de se retrouver avec des classes gigantesques, aussi illisibles que réutilisables. C'est pourquoi les principes SOLID ont été inventés. Robert Cecil Martin aussi connu en tant que l'Oncle Bob est un ingénieur en informatique et écrivain. Il a introduit de multiples concepts de programmation, décrivant et argumentant de bonnes pratiques de cet art, comme le fait de bien nommer les variables afin qu'elles explicitent leur utilité et les valeurs qu'elles stockent, ou de ne pas faire de copier-coller modifiés, mais de plutôt écrire des fonctions réutilisables à appeler avec différents paramètres pour faire varier un comportement. Dans la même logique de cohérence et d'uniformisation, il énonça les cinq principes fondamentaux de la programmation orientée objet.

Simple responsabilité unique : une classe, une fonction ou une méthode doit avoir une et une seule responsabilité. **Ouvert / Fermé** : tout élément d'une application, que ce soit une classe, une fonction ou autre, doit être fermé à la modification, mais ouvert à l'extension. Le développeur doit pouvoir grâce à son interface de programmation (API), ses paramètres ou sa configuration, l'utiliser voire l'interconnecter à un autre logiciel. **Liskov et sa substitution** : Une instance de véhicule doit pouvoir être remplacée par une instance de voiture, tel que la voiture est un sous-type de véhicule, et ce sans que cela ne modifie la cohérence du programme. **Interfaces séparées** : Il vaut mieux créer différentes interfaces spécifiques quitte à ce qu'elles n'aient qu'une seule méthode ou fonction, plutôt qu'une seule interface générale en possédant plusieurs. Cela permet d'avoir une granularité plus fine et de pouvoir obtenir des interfaces composées uniquement des méthodes nécessaires à l'aide de la composition permise par l'héritage. **Dépendances inversées** : Bien que les classes dépendent d'abstractions telles que les interfaces, il faut le plus souvent possible utiliser directement les interfaces de plus haut niveau. C'est-à-dire que je dois par défaut gérer des véhicules, et au besoin je change de filtre pour avoir toutes les informations voulues sur ma voiture, mon bateau ou autre. Il faut partir du concept général pour traiter le plus de cas et ensuite aller vers le particulier au besoin.

Problèmes et solutions

Lorsqu'un développeur ne trouve pas intuitivement comment implémenter la logique de façon algorithmique, il se retrouve souvent sur des forums de discussion comme StackOverflow ou désormais sur des outils conversationnels de génération de texte comme ChatGPT. De manière générale, il ne sert à rien de réinventer la roue. Si quelqu'un a la solution, autant juste l'utiliser. C'est d'ailleurs pourquoi les cadriciels et outils logiciels en tous genres sont aujourd'hui indispensables. Il est également important d'utiliser les outils logiciels adaptés. En JavaScript, si le fonctionnement d'un code est obscur, il est possible d'utiliser l'instruction `debugger;` pour indiquer un point arrêt. Lorsque l'interpréteur JavaScript va traiter cette instruction, il va

marquer une pause, ouvrir la console développeur sur le fichier contenant cette instruction, et fournir au développeur la possibilité de savoir qu'elles sont les valeurs contenues dans ses variables à ce moment `t`, en plus de lui permettre d'exécuter les instructions une après l'autre.

Une fois que nous avons identifié le problème, il convient d'analyser la structure du code et de l'améliorer pour obtenir la logique voulue. En programmation-objet, il existe des **patrons de conceptions** qui trouvent leurs racines dans l'architecture. L'architecte en bâtiment Christopher Alexander publie l'essai *A Pattern Language : Towns, Buildings, Construction* en 1977. Cette œuvre a défini de manière casuistique, à la manière du [Code de Hammurabi](#), les problèmes rencontrés en architecture conjointement à leurs solutions. Cette notion a permis de catégoriser les patrons algorithmiques selon trois classes.

Les *créateurs* solutionnent des problèmes d'[instanciation](#) et de configuration des classes et des objets. Les *structuraux* indiquent comment organiser les classes d'un programme et leurs interfaces. Et les *comportementaux* définissent l'organisation des [objets](#) pour que ceux-ci s'échangent des informations en fonction de leurs responsabilités.

Maintenant que nous avons le comportement voulu, nous pouvons rencontrer des problèmes de performances et d'optimisations. Pour les régler, il est possible de répartir la puissance de calcul entre plusieurs cœurs de processeurs à l'aide de sous-processus (voir fils / threads).⁴⁰ Nous pouvons aussi la répartir entre différentes machines, c'est-à-dire faire du calcul distribué. De manière générale, lorsque nous découpons une tâche en sous-tâches réalisables par des acteurs distincts, nous réduisons la charge de travail grâce à une logique de parallélisation.

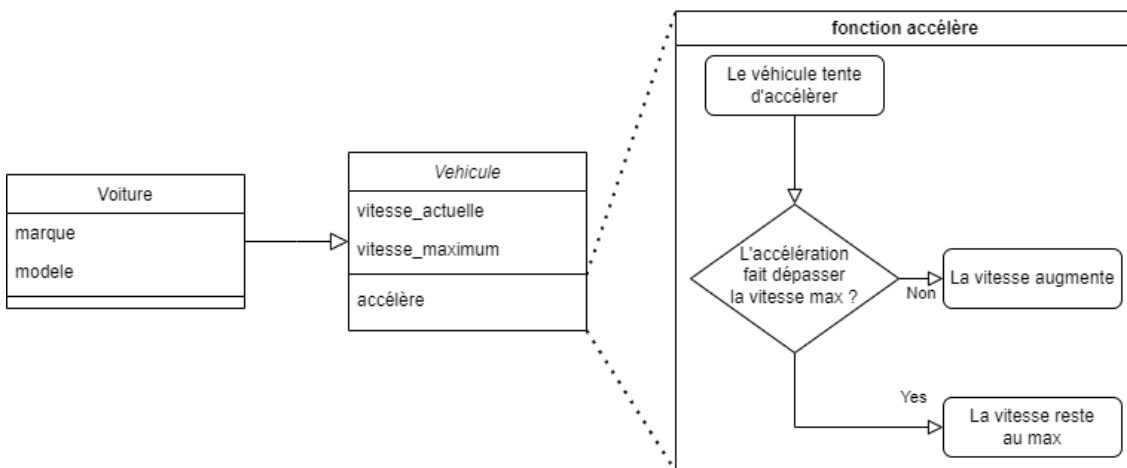
La gestion permet donc d'optimiser les charges et temps de travail. Une fois que le projet est abouti, il faut le tester et s'assurer que la qualité du produit est conforme aux exigences et besoins utilisateurs. Dans une grande partie des lignes de production, il y a une équipe dédiée aux tests et à la vérification. Elle est chargée de s'assurer que le produit est conforme à leurs attentes ou à celles formulées dans un document dédié comme un cahier des charges idéalement technique. Dans le monde de la production logicielle, nous avons la chance de pouvoir l'automatiser à l'aide de pratiques DevOps. Intégration et déploiement continu (CI / CD) permettent non seulement de vérifier le bon fonctionnement d'une liste de fonctionnalités, mais également de les mettre automatiquement à disposition du public lorsqu'elles sont toutes valides. Assez de code pour le moment. Je suis de l'avis que ces abstractions devraient être représentées et manipulables de manière graphique, de la même manière que le reste. La syntaxe devrait être optionnelle comme c'est le cas dans l'éditeur Typora avec lequel j'ai rédigé ce mémoire en code Markdown.

UML

Les flowcharts, qui se traduisent en français par arbres de décisions, diagrammes de processus, logigrammes ou encore organigrammes de programmation (et non pas par diagrammes de flux), ont initialement été développés dans l'ingénierie industrielle vers 1920.

Ils ont rapidement été repris par l'industrie informatique alors naissante vers 1950. La réalisation de schémas a, depuis les débuts de l'informatique, permit de formuler la logique la constituant pour modéliser les programmes qui en résulteraient, dans le but de les présenter à toute personne sans avoir à écrire ou lire du code.

C'est ainsi qu'en **1996**, Grady Booch, James Rumbaugh et Ivar Jacobson, trois ingénieurs travaillant sur des langages de modélisation objets différents, décident de s'unir pour co-crée un Langage de Modélisation Unifié, qui avec l'inversion anglaise donne l'acronyme UML. Reprenons notre véhicule, la fantastique voiture ne pouvant qu'accélérer. Nous pouvons la modéliser ainsi en UML.



Graphique UML discrétisé d'une voiture en tant que véhicule

Bien que ce schéma représente le code que j'ai écrit plus haut, il fait cependant abstraction de toute la syntaxe textuelle propre au code, et n'offre ainsi pas une interface d'édition en direct. Ce que je propose en tant que réponse à la problématique, c'est de représenter visuellement les fichiers de code en conservant l'ordre des instructions tout en facilitant sa visualisation.

Comment repenser la gestion de l'information pour moderniser l'expérience développeur ?

La programmation est une pratique récente dont les prérequis techniques ont mis longtemps à se mettre en place et évoluent depuis à une vitesse exponentielle. Là où les mathématiques ont permis de quantifier le monde, l'informatique peut quant à elle en modéliser une représentation au travers d'une simulation informationnelle. En tant qu'artisans du code, créer une simulation revient à fournir une interface textuelle, visuelle, et sonore. Cela constitue ainsi un jeu vidéo, même si celui-ci ne consiste qu'à être témoin d'une scène. Les créateurs et/ou inventeurs des différentes interfaces hommes-machines (IHM) s'efforcent de réaliser de telles scènes, usant de l'informatique pour imiter la nature, notre environnement ou notre histoire. La réalité virtuelle (VR) est une interface récente nous permettant une immersion totale. Elle se démocratise lentement à raison de 85 millions d'utilisateurs en 2022 (virtuelle et augmentée confondue — source : Statista⁴¹). La réalité, qu'elle soit réelle, virtuelle, augmentée, ou mixte, se trouve être un outil industriel intuitif et naturel aussi intéressant commercialement que pédagogiquement. Mais ce genre de matériel est actuellement un luxe trop peu répandu pour espérer avoir une adhésion facile et encore moins un large public d'utilisateurs. C'est pourquoi je me suis basé sur mes compétences en technologies web actuelles. Compétences que j'approfondirai à terme en réécrivant mon projet existant en Rust, après mes études à l'École Hexagone et l'obtention d'un prototype accessible sur internet. Pour l'instant, je vais me contenter de vous présenter les fondations de ce projet que j'ai rapidement introduit [en préface de ce livre](#).

Toute personne étrangère à la programmation, arrivant dans un projet logiciel via un IDE actuel, sera naturellement rebutée par toutes ses lignes de textes, qu'elles soient de commande ou de code. L'expérience développeur est aujourd'hui bien plus proche du bloc-note que des interfaces intuitives et agréables, voire fun à utiliser. Les développeurs travaillent sur une technologie futuriste dont l'usage est archaïque. Les outils no-code fleurissent ses dernières années. Ils permettent certes la réalisation de produits plus rapidement qu'en partant d'un cadriciel et en le codant soi-même. Cependant, dès que l'utilisateur désire un rendu différent de ce qui est proposé par le service, cela provoque un blocage, nécessitant l'intervention d'un développeur et de code. Un service proposant à la fois des outils visuels et des éditeurs de code sera alors considéré comme low-code. Ma solution se rapproche plus de ce mode de fonctionnement, car mes interfaces affichent le code, mais les mots clés et la syntaxe qui le constituent seront représentés par des symboles. Seuls les concepts clés et la logique auront besoin d'être appris par l'utilisateur. Mon motif directeur dans le développement d'EditIDE a été, est, et restera toujours de simplifier au maximum son rétrofonctionnement, tout en épargnant à ses utilisateurs les particularités des langages utilisés. L'interface doit se décrire elle-même comme un personnage se présenterait à autrui. L'utilisateur doit, quelle que soit la page du logiciel sur laquelle il est, comprendre ce qu'il peut faire, et trouver ce qu'il cherche. Allant toujours des concepts globaux aux cas précis.

Derrière la question fondamentale de ma problématique résident plusieurs questions annexes auxquelles je vais d'abord répondre. À commencer par « *Pourquoi la programmation est-elle aussi peu démocratisée ?* ». Soyons franc, tout le monde n'a pas besoin de savoir coder. La meilleure qualité d'un codeur n'est d'ailleurs pas de savoir coder, c'est sa capacité à proposer une solution scientifique et logique à un problème de donnée. C'est pourquoi je vais répondre à cette question avec quelques informations statistiques. En 2013, 1 personne sur 623 était développeur professionnel, soit 0,16% de la population mondiale. 1 sur 387 savait alors coder, soit 0.26%. En novembre 2022, la population a dépassé les 8 milliards d'habitants, celle des développeurs représentait alors 27.7 millions d'individus, soit 0.35%. La même année, 40 % des recruteurs prévoyaient d'embaucher 50 développeurs ou plus. L'étude liée à cette information note que le pourcentage de ceux qui recrutent de 201 à 500 personnes a doublé par rapport à 2021. Le problème le plus notable est dans le web où il y a 60 postes disponibles pour 38 demandeurs d'emploi. C'est pourquoi 57% des recruteurs se déclarent prêts à se passer du CV lors du processus de recrutement. Le pourcentage de recruteurs qui engagent des développeurs sans bagage académique a presque doublé (de 23% en 2021 à 39% en 2022).⁴²

Il y a donc des opportunités immenses pour les autodidactes et les écoles en informatique. Mais cette discipline requiert des connaissances avancées, qui ne sont pas enseignées lors de l'instruction obligatoire, parfois en option au lycée. C'est également une discipline en constant changement, qui requiert beaucoup de veille et de pratique. C'est pourquoi il est compliqué d'être dévoué à ce domaine, que nombre de personnes quitteront pour quelque chose qui leur convient mieux. Les développeurs expérimentés sont donc logiquement très recherchés, car un développeur junior mal supervisé peut être source de bugs et donc d'insatisfaction client.

En réalité quelle est l'origine des bugs ? À l'exception d'une erreur matérielle venant corrompre la mémoire ou interférer avec un signal, une machine en état de marche ne fait que ce que son programme lui indique. Les bugs sont donc presque toujours dus à une erreur humaine. Cette dernière vient soit d'un besoin initial mal compris, transmis, ou implémenté ; soit d'un enchaînement d'actions causant des effets de bords et venant modifier des fonctionnalités existantes, pouvant devenir incompatibles avec les nouvelles demandes et besoins. Les erreurs fatales viennent généralement de problèmes de mémoire ou de langages non compilés, qui n'ont donc pas de programme obligeant l'absence d'erreur avant usage. Et en tant que développeur, je peux vous dire que si le processus ne force pas les utilisateurs d'un outil à faire les choses correctement et corriger tous les potentiels problèmes, ils vont juste s'accumuler et former de la dette technique. La donnée étant la matière première d'un système d'information, elle peut être à l'origine de bugs. Par exemple, si la machine attend une donnée numérique et qu'elle obtient une donnée textuelle à la place, il faut gérer le cas, autrement il y a fort à parier que cela provoquerait un plantage, car il n'est pas possible d'additionner un nombre avec une cacahuète. Il convient de tester la robustesse de ses fonctionnalités en passant toute sorte de données de différents types et longueurs sans jamais que le programme ne plante. Dans la réalité, les équipes logicielles ne testent souvent que les cas qui marchent, et le jour où

le format de donnée change, les développeurs corrigent le plantage avec un patch, telle une rustine sur une chambre à air, sans prendre en compte les autres usures de cette dernière.

Avec la réalisation de ma solution, je souhaite automatiser et ludifier le processus de création d'une application ou d'un site web en JavaScript. Ce que j'ai fait, tout le monde peut le faire, et comme l'humain ne comprend quelque chose que lorsqu'il arrive à le réexpliquer plus simplement, à le factoriser pour rendre une information usinée, et bien je vais vous présenter ce qu'est concrètement EditIDE. Maintenant que j'ai détaillé l'infrastructure réseau nécessaire avec la mise en place d'un serveur NGINX, la création de certificats SSL, l'ouverture des ports HTTPS et la configuration d'un nom de domaine, je vais désormais répondre à la question « *Comment architecturer un système d'information ?* » en expliquant la structure de fichiers et de dossiers que j'utilise.

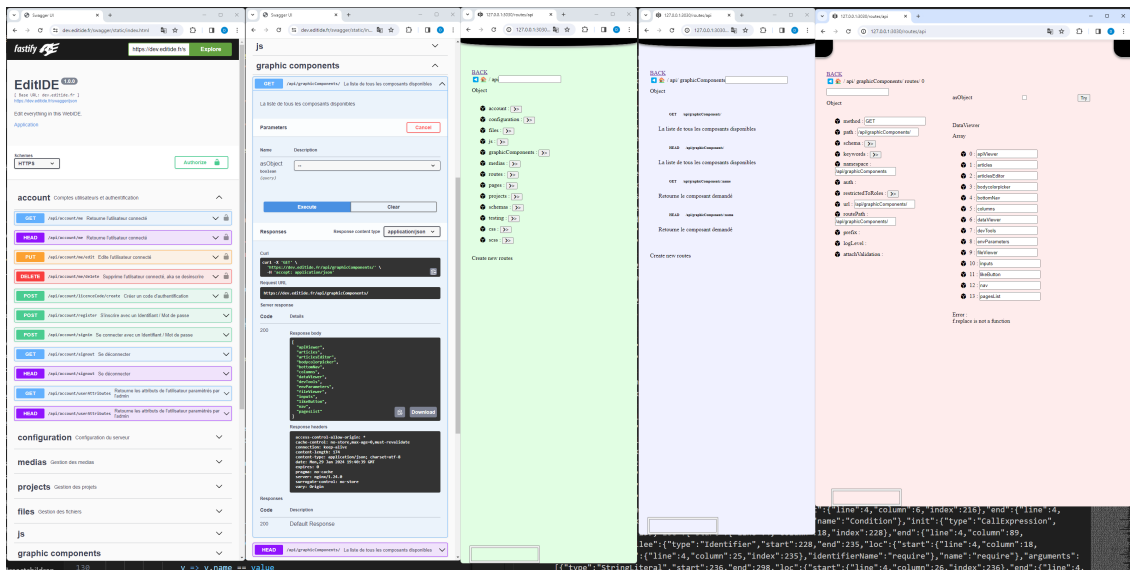
À la racine d'un projet logiciel se trouvent toute sorte d'outils, qu'il convient d'organiser et de séparer dans leurs dossiers respectifs. Pour EditIDE, j'ai fait le choix d'utiliser **Cypress** pour automatiser mes tests en ayant une interface WYSIWYG ne permettant malheureusement pas l'édition, bien que fournissant des outils de sélection des éléments du site web à tester. J'ai donc un dossier `cypress` contenant toutes les informations qu'il lui faut pour simuler le comportement d'un utilisateur final. J'ai de la même manière un dossier `docker` que j'ai implémenté pour m'assurer que si le système de la personne voulant installer le projet possède des incompatibilités, alors il peut l'exécuter dans un conteneur. Cela permet également de faciliter l'installation, le seul prérequis devenant alors Docker. EditIDE étant un projet JavaScript, à sa racine se trouve logiquement le dossier `node_modules` qui rassemble tous les outils logiciels tiers dont il dépend. J'ai aussi les dossiers `documentation` et `userData`, contenant respectivement les données que leurs noms indiquent. Comme sur tous mes périphériques de stockage, j'organise mon dossier `media` en fonction du type de document. Les quatre types principaux et donc sous-dossiers étant **texte**, **images**, **audio**, et **vidéo**. Enfin, le septième et dernier dossier à être situé à la racine est `code_source`, il est toujours explicite et contient ce qui suit : Le code du `model` de donnée, constitué de classes et de schémas de validation JSON. L'`admin_interface` décrivant la structure, le style et les interactions des pages web. Mes plugins du cadriciel Fastify, que je range dans le dossier `fastify-plugins` en attendant de les publier sur le gestionnaire de paquets NodeJS (npm) et ainsi pouvoir les installer comme dépendances. Et pour finir, le dossier `server`, correspondant au point d'entrée de mon application logicielle. Les scripts qu'il contient permettent de lancer le serveur avec différents paramètres, plugins et configurations, en fonction de l'environnement ciblé ou des tests automatisés. Le découpage de mon application en plugin est crucial, chacun contient son propre `model`, ses `services` ainsi que ses « handlers », « contrôleurs » ou « routes » définis dans le fichier `plugin.js` situé à leur racine, qui définissent comment traiter une requête. Chaque plugin correspond à une famille de fonctionnalités que je vais désormais présenter.

A priori

Je vais en priorité vous parler des fonctionnalités a priori, c'est-à-dire celles que j'ai actuellement réalisées et triées par modules. J'ai naturellement implémenté les utilitaires de gestion de tout serveur d'application web : comptes (authentification et permissions), compilateurs (gestions de langages comme pug ou scss), configuration, documentation, gestion de fichiers, composants graphiques, JavaScript, medias, base de données JSON (Mongo), pages, projets, routage, schemas (validation de donnée), shells (accès en ligne de commande), boutique e-commerce, style et thèmes, testing, et websockets (communication temps réel pour jeux, chats et autres). Avec ces modules j'ai pu réaliser différents projets d'interfaces graphiques et par conséquent d'applications. Je vais essentiellement vous présenter ceux qui répondent à la problématique, à savoir ceux qui facilitent déjà mon développement.

La documentation permet en théorie de répondre à toute question qu'un individu pourrait se poser concernant un domaine, ou dans le cas de la programmation informatique, d'un projet logiciel. Elle doit couvrir toutes les actions que doit accomplir le développeur, de l'installation du logiciel, à son lancement, son usage, ses fonctionnalités, son code, ainsi que comment contribuer et collaborer. Malgré le fait que je travaille seul sur ce projet, je suis souvent reparti de zéro pour réexpérimenter ce que vivrait un nouveau collaborateur. Cela m'a permis de rédiger des instructions claires et de me rendre compte de tous les problèmes qui peuvent être rencontrés pour y proposer les solutions associées. Le respect de la documentation, autant dans sa rédaction que dans sa lecture, permet d'améliorer la qualité et la sécurité de l'application.

Sur un serveur web, les fonctionnalités sont disponibles au travers des APIs. Étant donné que ce sont des sortes de boîtes noires, il est essentiel de les documenter. Pour ce faire, un des outils les plus utilisés est actuellement **swagger**. Sa fastidieuse configuration permet de regrouper des ressources web, identifiables par un verbe HTTP et une URL, dans des sections. En les dépliant, j'obtiens un formulaire permettant d'indiquer quels sont les paramètres et réponses attendues. Dans ma solution, je souhaite qu'il n'y ait pas besoin de renseigner ces informations, car je les ai en théorie déjà lors de la conception du modèle de donnée. De plus les documentations sont rarement mises à jour. Je souhaite donc les générer automatiquement en fonction du code, pour qu'elles soient toujours représentatives de la réalité, car fortement liées. Finalement, swagger s'apparente plus à une immense liste tandis que mon gestionnaire d'API est plus proche du gestionnaire de fichiers, avec une recherche qui fonctionne telle une ligne de commande, et une interface ne présentant que la ressource nécessaire autant en termes d'URL, de fonctionnalités ou de donnée. Lorsque je fais face à un problème informationnel, il convient de définir son périmètre, pour analyser et gérer précisément la zone voulue. L'informatique nous permet désormais d'avoir une interface pour naviguer précisément dans une simulation réaliste et visualiser ses ordres de grandeur. C'est pourquoi j'applique les grands principes du [c4model](#). Cette modélisation utilise des graphiques pour proposer une cartographie d'un projet logiciel sous forme de contextes, conteneurs, composants, et code. Cela permet une vue macro et micro, précise à la granularité voulue.

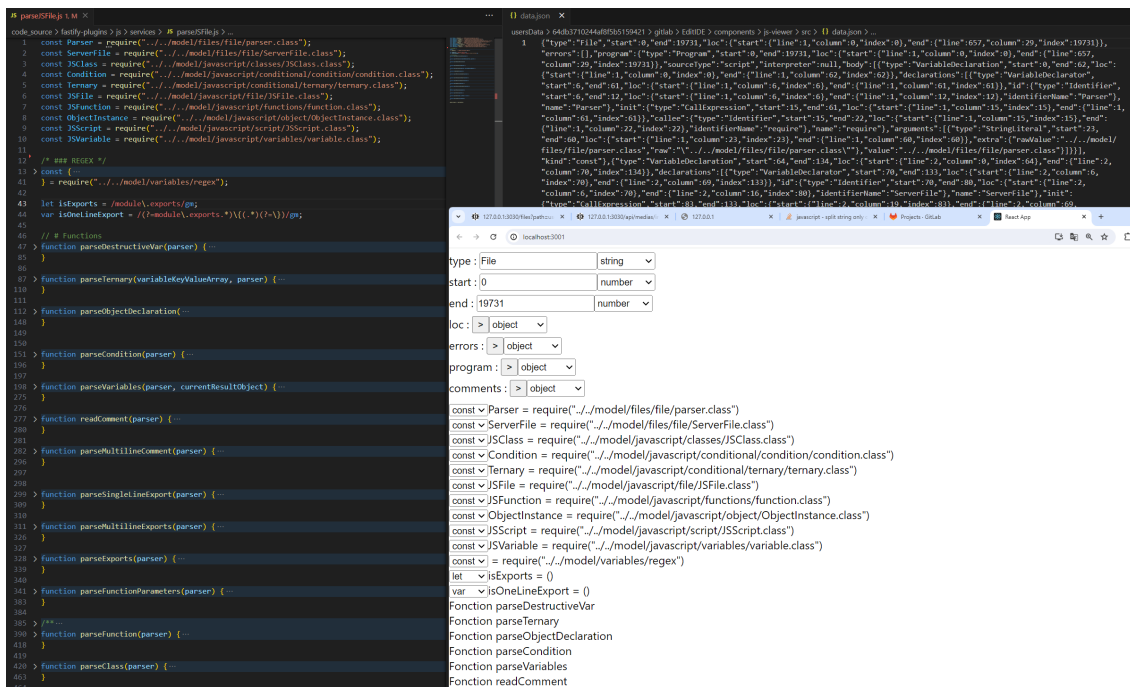


F11 : Comparaison d'interfaces entre celle de Swagger et ma proposition de système

Comme vous pouvez le voir dans l'illustration ci-dessus, mon gestionnaire d'API permet de tester les fonctionnalités comme sur swagger. Dans les deux outils, l'utilisateur peut renseigner ses informations dans un formulaire pour utiliser les fonctionnalités du site. La différence est que celui d'EditIDE est autogénéré en fonction du code JavaScript, et que le développeur pourra ultérieurement le paramétrer sans utiliser de code, directement depuis l'interface. L'autre différence notable est que l'affichage des données renvoyées par mon serveur EditIDE n'est pas représenté sous forme de texte, mais sous celle d'un explorateur semblable à un gestionnaire de fichiers. Une API renvoie souvent un format correspondant à un objet, dans notre cas, du JSON. Des outils de visualisation du JSON existent, mais ils ne me conviennent pas, car ils se contentent de l'afficher tel quel, en temps que long texte brut incompréhensible à sections pliables, ou toujours en affichant autre chose que la ressource au chemin actuel. Dans mon affichage, je ne souhaite n'afficher que les clés et valeurs du chemin où je suis actuellement, à commencer par la racine de mon contexte. De la même manière, quand je travaille sur une fonction je n'ai pas besoin de voir autre chose que ses dépendances. Tout le reste n'est qu'une information parasite. C'est pourquoi j'ai appliqué la même logique d'arborescence pour visualiser les fichiers de code, et je la garderai dans tout le fonctionnement d'EditIDE de la même manière que les fils d'arianes (Accueil > Page > Article > ...) sont devenus universels sur la plupart des sites web et applications.

Lors de mon développement de cette solution, j'ai rapidement eu besoin de créer un programme qui puisse lire ses propres fichiers de code et les représenter de manière abstraite. De la même manière que j'ai refait plusieurs fois le processus d'installation de mon logiciel pour le documenter, j'ai refait plusieurs fois cette fonctionnalité. C'est la première que j'ai implémentée lors de KIDE, ma preuve de concept (**Proof Of Concept**) réalisée en 2019 dont j'ai parlé en préface. J'ai nommé ce composant d'interface graphique le `js-viewer`. Pour l'instant il affiche juste un fichier de code JavaScript comme si j'avais replié toutes ses fonctions dans mon éditeur de texte. Cela permet d'avoir une bonne visualisation de l'ensemble

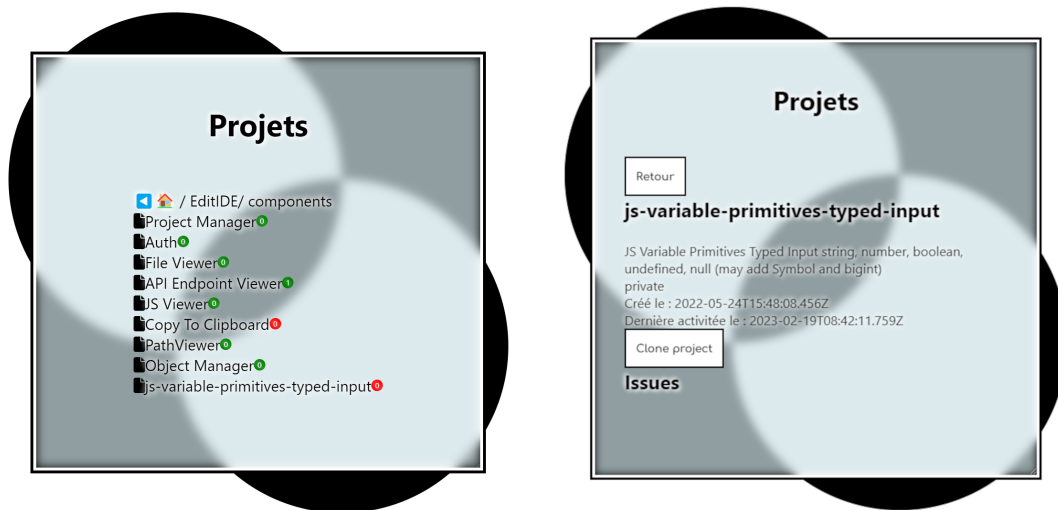
des imports, variables, et fonctions du fichier. Dans mon POC de 2019, je lisais moi-même le fichier pour générer un Object JSON qui le représente. Je me suis rendu compte par la suite que les outils comme Babel, qui permettent entre autres de compiler du TypeScript en JavaScript, font exactement cela en créant ce qui s'appelle un Abstract Syntaxic Tree ou AST. Comme vu précédemment et comme l'acronyme AST l'indique, un fichier JSON est un arbre, ou un graphe, qu'il est possible de parcourir. C'est pourquoi je l'affiche telle une arborescence de fichier, en réutilisant mon composant React ObjectViewer. Il me permet de visualiser et de modifier des objets JSON. Dans le but de faciliter mon travail sur de tels projets d'interface annexes, j'ai également développé une interface permettant de gérer les données venant de mon plugin `files`. Cette dernière inclut le `js-viewer` et le `object-manager` pour pouvoir afficher les fichiers correspondants à leurs types pris en charge. Lorsque je suis dans un `package.json`, je peux ainsi exécuter les scripts qu'il contient, recompiler mes projets, voire même en extraire les fichiers. Tout cela en un seul clic, et ultérieurement de manière automatique.



F12 : Comparaison VSCode vs EditIDE

Je ne vais pas présenter tous les projets, mais face au grand nombre d'entre eux que je dois gérer et maintenir à jour, je me suis créé un `project-viewer`. Connecté à Gitlab à l'aide d'un token me permettant d'accéder à son API, il me permet d'afficher la liste de mes projets à l'aide à l'aide de mon `object-manager`, que j'ai configuré pour qu'il récupère un projet lorsque je clique dessus. Je prévois également d'ajouter une sélection et/ou d'ajouter un bouton pour récupérer tous les projets qui ne sont pas encore téléchargés. Gitlab n'affiche pas d'arborescence des projets. Par défaut la page d'accueil liste tous les projets par ordre alphabétique, et le menu latéral propose un lien pour voir les groupes et sous-groupes dans une page à part. C'est pourquoi j'ai préféré me faire mon propre affichage qui récupère d'abord tous

les projets, puis utilise uniquement les informations nécessaires pour les afficher sous forme d'une arborescence. J'ai par la même occasion commencé à implémenter les issues Gitlab. Cela m'offre un système de gestion de ticket déjà populaire et gratuit, qui me permet d'avoir le strict nécessaire en termes de gestion de projet.



F13 : Prototype project-manager utilisant l'API de Gitlab

Git est un logiciel de gestion de versions décentralisé sous licence GNU GPL. C'est un logiciel libre et gratuit, initié en **2005** par Linus Torvalds, le créateur de Linux, lors de la création de son noyau. Linus ne voulait pas réaliser ce projet seul et a rapidement compris que la qualité d'un logiciel viendrait des débats houleux entre ses collaborateurs et l'implémentation de la meilleure idée à la fin. J'ai de la même manière commencé à implémenter une gestion graphique de cet outil, notamment pour récupérer la dernière version à jour. À terme je la couplerai aux tests automatisés pour proposer à l'utilisateur de « sauvegarder » son code en faisant des commit dès ses modifications passent les tests. Je vais désormais vous présenter les fonctionnalités que je n'ai pas encore implémentées.

A posteriori

Plus un projet a d'intermédiaires, plus il y a de risques de déformations ou pertes d'informations et d'erreurs. Dès [le fondement de l'informatique](#), Charles Babbage l'avait déjà compris et sa machine faisait à la fois le calcul et l'impression, car beaucoup d'erreurs arrivaient lors de l'impression. En informatique, beaucoup d'erreurs ocurrent par manque d'information ou de clarté dans leur communication. Le client exprime parfois mal son besoin, voire a un problème informationnel qu'il ne saurait exprimer. Les équipes faisant l'intermédiaire entre le client et l'équipe de production se retrouvent souvent entre deux eaux et doivent s'adapter en apprenant deux jargons au vocabulaire très différent. Idéalement ils devraient avoir été, ou du moins être capables de se mettre à la place des deux partis dont ils font l'intermédiaire, dans le but de mieux les comprendre et pouvoir efficacement

communiquer avec eux afin de trouver la meilleure solution. Concernant la gestion de projet informatique, des bugs et des incidents, que ce soit une nouvelle fonctionnalité ou un problème rencontré dans le logiciel, il convient d'énoncer clairement le problème afin que toute personne profane puisse le comprendre et le réexpliquer. C'est pourquoi dans le cas d'un bogue rencontré dans l'application, je pense que l'idéal serait que l'utilisateur puisse remonter la suite d'action problématique ou attendue au serveur, qui la convertira automatiquement en test d'interface (E2E). Je souhaite automatiser ce processus dans EditIDE pour pouvoir facilement recueillir et traiter les bugs en m'assurant que le comportement restera correct après résolution.

For an individual, however, there can be no question that a few clear ideas are worth more than many confused ones.

Charles S. Peirce—*How to make Ideas clear*

Certains problèmes arrivent lors de l'échange entre l'équipe conception (design) et l'équipe technique. Il arrive que des concepts soient validés alors qu'ils ne sont tout bonnement pas réalisables en termes de code, généralement due à des limitations CSS, ou des problèmes informationnels dus à des cas qui n'ont pas été pensés, comme la longueur des textes pouvant varier, l'architecture du modèle de donnée, ou le changement de sens de lecture et d'écriture dans certains langages comme l'arabe ou le mandarin. De plus, pour la plupart des projets, plusieurs développeurs doivent collaborer et travailler ensemble sur le même code, ce qui peut entraîner des conflits de version, des problèmes d'intégration et de coordination. Tous ses problèmes arrivent car rien n'est centralisé ou correctement documenté, à une heure où la collaboration est mise en avant comme jamais. Dans EditIDE, tout se passerait sur le site en lui-même. La conception et l'intégration ne feraient désormais plus qu'un, les développeurs front ne seraient chargés que de dynamiser les concepts déjà intégrés avec les appels API du back qui seraient autogénérés. Idéalement, une telle solution doit donc satisfaire l'ensemble des acteurs d'un projet informatique. C'est pourquoi cette application sera éditable depuis sa propre interface, qui pourra donc s'éditer elle-même. Comme dans un jeu, il sera possible de sélectionner un élément graphique pour voir uniquement ses informations, et les actions réalisables avec. Si cet élément a été généré, il sera possible de retrouver son composant contenant sa structure, son habillage et ses interactions, voir l'objet auquel il correspond. Ainsi EditIDE fournira autant des outils aux webmasters qu'aux designers.

L'informatique et l'algorithmique nous permettent de répliquer à l'infini une information ou un traitement logique de donnée. Cependant le code doit être unique, le dupliquer est donc un problème, et générer une fonction avec des paramètres pour configurer sa logique, la solution. Un de mes objectifs sera donc de proposer une interface permettant de le faire systématiquement. De manière générale, l'interface relie un élément graphique à une fonctionnalité. Un utilisateur averti sait qu'il a sur son clavier des touches de modification comme **CTRL**, **SHIFT**, ou **ALT** permettant d'appliquer des effets ou réaliser des sélections multiples. Il peut même passer d'un élément au suivant avec la touche tabulation, cocher ou

décocher une case avec espace, valider avec entrée, supprimer avec la touche retour ou celle du même nom. Bien que je compte réaliser des interfaces pour programmer et gérer la réalisation d'outils logiciels, je conserverai bien évidemment l'utilisation des raccourcis clavier et des commandes textuelles. Cependant, les actions que l'utilisateur peut ou non réaliser doivent être visuellement explicites, et indiquer les raccourcis clavier ou commandes correspondantes. Tout utilisateur doit pouvoir facilement créer de nouvelles instructions sans devoir choisir dans une liste d'autocomplétions interminable remplie d'éléments impertinents. L'intelligence artificielle pourra m'aider à améliorer ce problème de pertinence, mais le principal désagrément vient selon moi de l'amoncellement de fonctionnalités proposées en même temps à l'écran, incluant les longs affichages de texte brut. Vous l'avez sûrement déjà compris lors de ma présentation des fonctionnalités existantes, mais quand j'arrive dans un fichier de code, je veux savoir directement le nom des variables et fonctions qui le constitue. Je me fiche d'avoir leur contenu ou détail, ni même leurs paramètres ou leur type de retour ; bien que je dois pouvoir rechercher en fonction de ces derniers. De la même manière, quand je travaille sur une fonction je ne désire voir que ce qui concerne cette fonction. Il n'y aura pas moins de fonctionnalités pour autant. Comme sur une version mobile, l'interface deviendra minimaliste et s'adaptera à l'utilisateur. Souhaitant cela, je ferais en sorte que n'importe qui puisse comprendre le code. De même que chaque fonctionnalité doit être documentée, et idéalement 100% du code doit être couvert par les tests, l'interface incitera les développeurs à commenter chaque ligne de code en langage naturel. Un switch sera alors disponible pour basculer entre le code réel et sa version en langage naturel entièrement commenté. Tout manquement déclencherait une notification sur la ligne en question et une baisse du pourcentage de couverture. Si je pense en tant que chef d'entreprise, je me fiche du langage que tu utilises, fais-le en patate ou en carotte si tu veux, mais réalises les fonctionnalités demandées, interconnecte-les et fais en sorte que ça marche de manière pérenne. Je me fiche du temps que ça prend, quand tu auras fini ta tâche, documentée elle sera, par n'importe qui elle pourra être reprise, et optimisée tu l'auras. Enfin, testée sera l'application, et ce du point de vue de l'utilisateur.

Conclusion

En définitive, ce que je propose en tant que solution pour moderniser l'expérience développeur, c'est un outil logiciel qui me permet de mieux visualiser mon code et gérer mes projets. Et ce de manière direct-live, à l'intersection entre Scratch, Elementor, Typora et un Notebook Python. Plus besoin d'alterner entre différents applicatifs ou onglets pour travailler, de nos jours les technologies web sont prédominantes et accessibles. Elles font désormais partie des normes, qu'il convient d'utiliser et de suivre. La programmation ne restera pas cantonnée aux lignes de commande ni aux blocs-notes remplis de texte brut. Le code est actuellement si repoussant que des solutions no-code et low-code se présentent aujourd'hui comme une manière plus pratique et agréable pour automatiser des actions ou réaliser des applications logicielles. Elles sont pourtant souvent limitées à un domaine particulier ou à des fonctionnalités prédéfinies.

C'est pourquoi la solution que j'ai proposée consiste à rendre le code plus lisible. Attaquons-nous aux causes plutôt qu'aux conséquences. Simplifier l'interface des outils de développeurs permettra d'améliorer notre propre expérience en plus de celle de nos collaborateurs non techniques. En plus de l'expérience développeur, l'idée globale est de centraliser l'intégralité du processus de réalisation d'un site web et de sa maintenance, sur le site web lui-même.

En tant que développeur, je peux affirmer d'un point de vue technique que ce logiciel proposé en tant que modernisation de l'expérience développeur me permet déjà de faciliter mon utilisation des différents projets que j'ai interconnectés, et qu'il me permettra à court terme de récupérer n'importe quel projet JavaScript depuis mon Gitlab, Github ou n'importe quel dépôt. Je pourrai ensuite installer les dépendances de leurs `package.json` et exécuter les scripts qu'il contient depuis une interface appropriée. Après quoi je pourrais naviguer dans n'importe quel projet JS, visualiser et éditer son code, ses variables et fonctions. Et ce de manière plus simple, concise et efficace grâce à une API et une interface créée avec EditIDE pour EditIDE. Là est la force de la rétroaction et de la métaprogrammation. Loué soit [Babel](#) et [TSTC](#). Rust suivra.

Avant de finir ce mémoire, je tiens à répondre à une ultime question : « Quelles sont les conditions de succès d'un logiciel ? ». De manière générale, le succès d'un logiciel se fait si ce dernier est accessible, facile d'utilisation, et utile. L'expérience procurée par l'application doit dépasser les désirs, attentes et promesses faites en amont. Au-delà d'un logiciel, je souhaite réaliser un ludiciel éducatif professionnel. Un ludiciel doit être vivant, animé, amusant et surtout naturel. La notion de naturel est selon moi très liée à une logique autodescriptive. Le code lui-même suit une logique qui se décrit elle-même. C'est pourquoi mon objectif principal est de faire en sorte qu'il soit affiché avec des symboles autodescriptifs, couplé à une explication textuelle dans la langue de l'utilisateur si nécessaire. Comme tout jeu vidéo, ce logiciel devra fournir à l'utilisateur toutes les informations nécessaires à son autoformation. En outre, tout joueur doit pouvoir suivre son évolution et être récompensé pour sa progression.

Enfin, le succès d'une œuvre ne se fait que si les spectateurs y adhèrent et la partagent. La viralité est un facteur mathématiquement calculable. C'est la moyenne du nombre de retransmissions faites par les personnes recevant une information. Une information est par conséquent virale dès que son facteur dépasse 1, autrement dit dès que la moyenne des personnes en sa possession la retransmettent à plus d'une autre personne.

Je ne sais pas de quoi sera fait le futur. Je pense que je vais devoir l'écrire après le présent moment qui achève ce livre. Je n'aurais nullement la prétention de dire qu'EditIDE est le futur, car pour cela il faudrait que le logiciel existe déjà de manière utilisable et utilisée.

Si vous voulez participer à son succès, partagez ce livre avec deux personnes que vous pensez pouvoir y être sensible. En espérant que plus d'une finira par vraiment l'être.

Encore merci de m'avoir lu.

Postface : Rétrospective et métacognition

J'ai écrit ce mémoire en sous-marin, avec le même état d'esprit qu'EditIDE depuis maintenant 5 ans. J'ai été surpris que certains de mes amis découvrent le site web `bonjour.blanchardorian.fr` à travers une autre URL qu'ils connaissaient. Cela m'a rappelé que mon système est désormais assez protégé pour ne pas avoir été mis à terre par des botnets comme j'avais pu l'expérimenter en laissant une base de donnée accessible sur le port 26777, sans paramétrage d'authentification, c'est-à-dire en libre accès de lecture et d'écriture. Je n'avais alors rien dans ma base, mais si cela avait été le cas, tout aurait été chiffré par un rançongiciel.

J'ai écrit *Origines et Évolution de l'Informatique* alors que j'ai longtemps haï l'écriture, surtout manuelle, et je ne l'ai que rarement pratiquée même de manière numérique. Cependant, j'apprécie désormais savoir et pouvoir partager mes idées de manière claire et la plus explicite possible, reflétant que ce que je crois être scientifiquement et historiquement vrai. Quand j'écris ma pensée, je peux la relire comme si je m'entendais la prononcer, et ainsi la reformuler jusqu'à ce que son intention soit distincte. J'ai ainsi du mal à comprendre les philosophes antiques qui auraient dit ne pas vouloir transmettre leurs pensées à l'écrit, car elle ne peut alors plus se défendre elle-même. Bien que rappelons le, à cette époque l'imprimerie n'existe pas et encore moins la dactylographie, rendant la tâche plus complexe et rebutante.

J'ai écrit en commençant par de la prise de note puis en rédigeant, reliant petit à petit toutes les idées et concepts dont je voulais parler. En soi, comme beaucoup de personnalités évoquées dans mon œuvre, je n'ai rien inventé. J'ai découvert plein d'informations et me suis contenté de les compiler à la manière de Vitruve, ou d'un ordinateur. Chaque idée ou sujet que je voulais évoquer m'a fait parcourir livres et dizaines d'onglets Wikipédia, et demandé des heures de recherches et de lecture pour finir par rédiger un maigre paragraphe résumant le thème étudié aux informations que je trouvais les plus pertinentes. Dès que j'avais une idée ou une information qui pouvait se relier à mon mémoire, je dégainais mon téléphone, activais mon VPN pour me connecter au réseau de mon appartement et avait ainsi accès à un Windows à distance. Ce dernier tourne h24 sur un mini-pc à 500 euros que j'ai acheté avant d'intégrer l'École Hexagone, avec le premier salaire de mon alternance et la jolie prime obtenue pour avoir rejoint l'entreprise dès que possible un mois avant la rentrée scolaire. D'ailleurs, encore merci à ABGX, ses salariés, et ceux d'Hexagone, en postes lors de ma scolarité de 2022 à 2024.

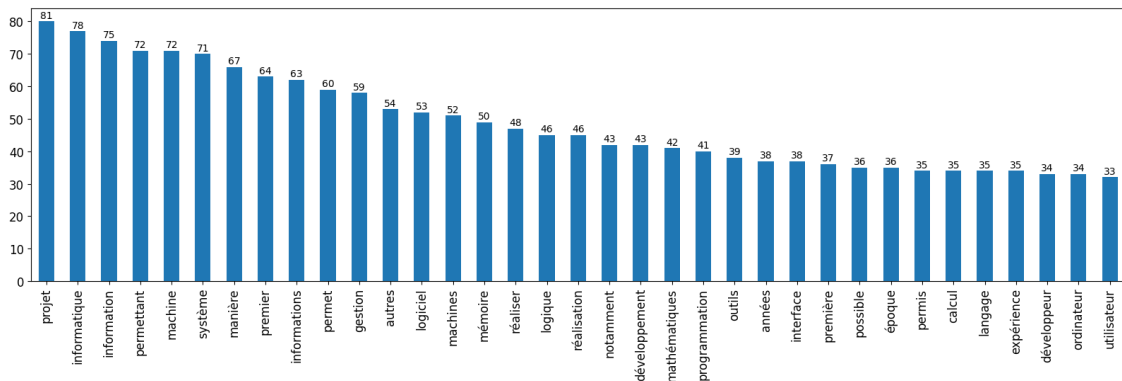
J'ai écrit ce mémoire assez naturellement, en le commençant en 4^{ème} année, avec l'idée globale de ce qu'un mémoire devait comporter, recueillie auprès d'élèves de 5^{ème} année lors des séances de sports, un an avant d'avoir eu l'ensemble des consignes données par M. Pachon. Je tiens d'ailleurs à remercier ce dernier pour ses consignes qui m'ont permis de mieux structurer mon document et de le valider.

Enfin, j'ai écrit (promis c'est le dernier), en pensant à tout lecteur, que je tiens à remercier affectueusement. J'ai moi-même relu une dizaine de fois ce pavé afin de m'assurer qu'il contienne le moins d'erreurs possible, autant dans les informations, que dans son orthographe et sa grammaire. J'espère que malgré la quantité, sa qualité aura su ravir quelques lecteurs.

J'ai parfois rencontré des problèmes pour retrouver des informations, pour [Raymond Lulle](#) par exemple. J'ai dû traduire la page catalane en français, puis croiser avec le livre *Histoire de l'informatique illustrée*, et ce afin de mieux comprendre le lien entre Lulle et la logique combinatoire qui a inspiré [Leibniz](#) pour la logique computationnelle. En effet, certains concepts comme l'*automatic programming*, ne possèdent pas de page Wikipédia ni de définition en français. J'ai donc très souvent croiser les versions anglaises et françaises des pages Wikipédia que je consultais ainsi que leurs bibliographies pour être sûr de la source et de la véracité des informations. Cela m'a permis de découvrir et lire des textes historiques fantastiques comme *School and Society* de John Dewey, *How to Make Our Ideas Clear* de [Charles S. Peirce](#) ou le texte *As we may think* de [Vannevar Bush](#), prouvant que beaucoup d'inventions et informations qui n'étaient pas connues du grand public, étaient déjà accessibles aux plus curieux.

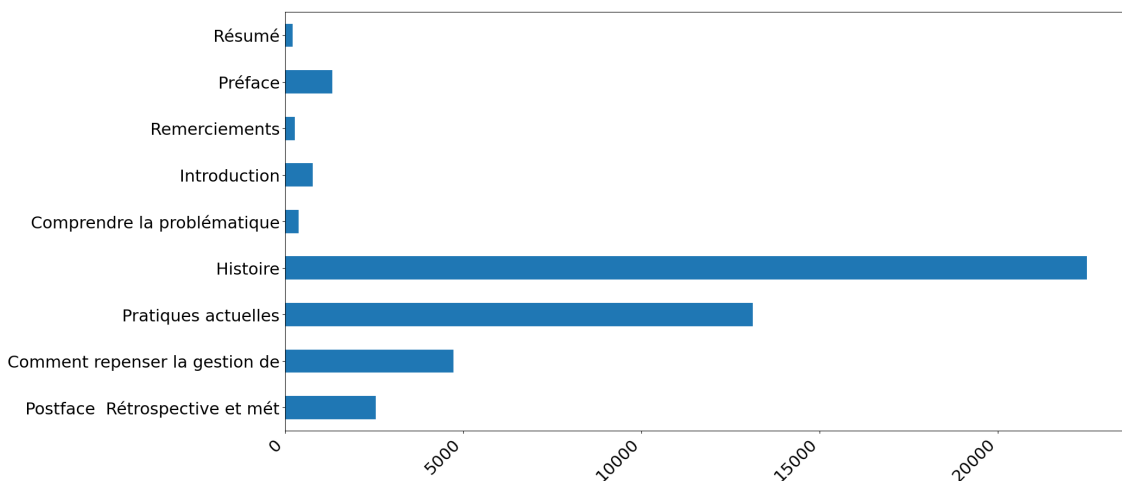
Je dus faire des concessions, et j'ai dû parler de personnalités qui ont dit, écrit, ou fait des choses regrettables, dont Rousseau qui disait que « *Toute l'éducation des femmes doit être relative aux hommes, leur plaire, leur être utile, se faire aimer, honorer d'eux, les élever jeunes, les soigner grands, les consoler, leur rendre la vie agréable et douce. Voilà les devoirs d'une femme dans tous les temps et ce qu'on doit leur apprendre dès leur enfance* ». Autres temps, autres mœurs, diront certains. Mais ce genre d'injonction à l'empathie ne devrait pas être genré au féminin, au risque même d'en priver les Hommes dans le sens de l'humanité. Nicole-Claude Mathieu a déclaré que la féminité est « *imposée aux femmes, et de la masculinité, apprise aux hommes* », et je souhaite à tout le monde d'apprendre sans jamais ne se laisser imposer quoi que ce soit contre son gré. Il faut parfois savoir battre en retraite, cela s'apprend au même titre que le courage, qui rappelons-le, n'est pas l'absence de peurs, mais sa capacité à les gérer.

Ce texte conséquent étant écrit en Markdown, j'ai pu facilement traiter sa donnée et l'analyser avec des graphiques.⁴³ La pratique du Big Data repose sur les 5 V de l'information, soit volume, vitesse, variété, véracité, valeur. Pour la rédaction de ce mémoire, le volume devait être contenu entre 80 et 85 pages d'entre 25 et 35 lignes, la progression du discours devait avoir une vitesse adaptée pour que le lecteur puisse suivre, les informations doivent être assez variées pour éviter la redondance, la véracité a aussi été au cœur de mes préoccupations lors de mes relectures de sources, et enfin chaque information doit ajouter de la valeur au texte.



F14 : Top 35 des mots de plus de 5 lettres les plus utilisés dans ce mémoire

Cette première analyse m'a permis d'être sûr que les mots les plus utilisés soient bien en rapport avec le titre, ou le sous-titre, c'est-à-dire le sujet. Ensuite, je me suis amusé à compter le nombre de mots de chaque partie pour m'assurer d'avoir un entonnoir allant du général vers le particulier. Et je suis personnellement satisfait de cette répartition qui m'a permis de traiter chacune des parties avec le strict nécessaire pour comprendre et résoudre la problématique.



F16 : Nombre de mots par chapitre de ce mémoire

Oui, j'ai utilisé Wikipédia et ChatGPT pour écrire cet ouvrage. Ce sont de précieuses ressources à utiliser avec précaution. Les intelligences artificielles pourraient ne pas avoir le droit d'auteur ainsi que d'exploitation commerciale. Je me prévaux donc de l'utiliser comme source sûre, et je les vérifie systématiquement au même titre que toutes les autres. Le passage sur Richard Stallman a néanmoins vu le jour grâce au modèle de génération de texte qu'est ChatGPT. Voici le dialogue qui en est à l'origine :

D

Qui dans le homebrew club a dit "My project is to make all software free" vers 1976 ?



En 1976, Richard Stallman a prononcé cette déclaration concernant son projet de rendre tous les logiciels libres. Richard Stallman est un programmeur et militant du logiciel libre qui a fondé le mouvement du logiciel libre dans les années 1980. Il a également lancé le projet GNU pour développer un système d'exploitation entièrement libre, et il est célèbre pour avoir développé la licence publique générale GNU (GPL), qui est largement utilisée pour distribuer des logiciels libres et garantir que les utilisateurs conservent les libertés essentielles du logiciel.

F17 : Identification d'un orateur présent dans le reportage « Révoltes et révolutions technologiques » d'ARTE

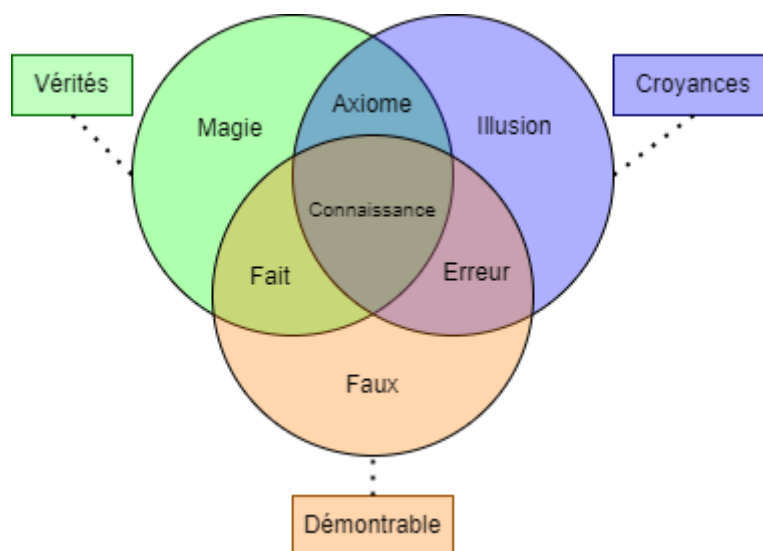
Cela m'a pris du temps de reprendre et aboutir mes études, d'autant que ça m'en a pris de relire des faits historiques et scientifiques. L'histoire est avant tout ce que l'humanité fait dans le *temps* que ce grand tout lui a accordé jusque qu'à présent. Comme dis au début de cette postface, hormis les relectures qui ont commencé mi-janvier 2024, c'est-à-dire 2 mois avant la date de rendu finale. J'ai été seul acteur de la relecture et de la rédaction de cette œuvre. Et s'il y a bien quelque chose que nous ne pouvons comprendre qu'en étant seul, c'est bien que devenir adulte conçoit avec faire tous les jours un peu plus attention au soi de demain. Tout réside dans la préparation et dans l'attention accordée à son soi-futur. Tout adulte a déjà été enfant, et ne le devient vraiment que lorsqu'il arrive à être à la fois son propre parent et enfant, mentor et apprenant. Puisque j'ai beaucoup travaillé sur l'écriture de ce mémoire qui m'a tant appris, je finirai sur une citation de [Célestin Freinet](#) parlant de l'imprimerie et plus généralement de l'écriture en tant qu'exercice pédagogique :

Dans la pratique, on ne se lasse jamais d'imprimer et les adultes se laissent prendre eux aussi à la minutie d'une technique qui permet la transcription en une forme magnifiée et définitive des textes auxquels on veut donner vie et harmonie. L'enfant qui compose un texte le sent naître sous sa main ; il lui donne une nouvelle vie, il le fait sien. Il n'y a désormais plus d'intermédiaire dans le processus qui conduit de la pensée ébauchée, puis exprimée, au journal qu'ils postent pour les correspondants : tous les échelons y sont : écriture, mise au point collective, composition, illustration, disposition sur la presse, encrage, tirage, groupage, agrafage. C'est justement cette continuité artisanale qui constitue l'essentiel de la portée pédagogique de l'imprimerie à l'école. Elle corrige ce qu'a d'irrationnel en éducation cette croyance que d'autres peuvent créer pour nous notre propre culture.

[Célestin Freinet](#), *Le journal scolaire*, 1967

Annexes

Mon œuvre peut être vendue avec mon accord, mais elle est libre et sera toujours accessible comme c'est le cas actuellement sur GitHub. Si vous voulez lire cela sur un livre rien ne vous empêche d'en payer un ou concevoir un vous même. Je vous laisse un schéma résultant du sujet initial de mon mémoire qui abordait plus en détail la philosophie et l'information de manière globale. Cela sera peut-être le sujet d'un autre ouvrage, qui sait !?



F18 : Diagramme de Venn pour classifier une information

Enfin, si l'envie vous en prend, bonne relecture. :)

Bibliographie

Computer, A History of the Information Machine — Martin Campbell-Kelly, William F. Aspray, Jeffrey R. Yost, Honghong Tinn, Gerardo Con Diaz.

The history of the computer : people, inventions, and technology that changed our world — Rachel Ignatofsky

Histoire illustrée de l'informatique — Emmanuel Lazard et Pierre Mounier-Kuhn

Transmettre — Christophe André, Céline Alvarez, Catherine Gueguen, Matthieu Ricard, Frédéric Lenoir, Ilios Kotsou, Caroline Lesire

Lean Startup — Eric Ries

Le livre des décisions — Mikael Krogerus & Roman Tschäppeler

ARPANET, le monde en réseau — Tristan Gaston-Breton

Webographie

Si une source n'est plus disponible, recherchez son URL dans sur <https://web.archive.org/>

Simon Parent : La programmation informatique à l'école primaire Pratiques effectives de programmation et mobilisation d'habiletés de résolution collaborative de problèmes (RCP) — Université de Montréal

Emmanuel Pietriga : Environnements et langages de programmation visuels pour le traitement de documents structurés. Interface homme-machine [cs.HC]. Institut National Polytechnique de Grenoble — INPG, 2002. Français. ffNNT : ff. fftel-00125472f

Man-Computer Symbiosis J. C. R. Licklider IRE Transactions on Human Factors in Electronics, volume HFE-1, pages 4-11, March 1960

Luciano Floridi : Philosophy and computing, an introduction

<https://web.archive.org/web/20230128142730/https://www.physique.usherbrooke.ca/~afaribau/essai/>

<https://www.enfant-encyclopedie.com/pdf/synthese/apprentissage-par-le-jeu>

<http://remacle.org/bloodwolf/erudits/Vitruve/livre1.htm>

<https://dl.acm.org/doi/pdf/10.1145/12178.12180>

<https://computingthehumanexperience.com/wp-content/themes/valeriya-child/timeline/acontimeline.html>

<https://fr.mathigon.org/timeline>

<https://passerelles.essentiels.bnf.fr/fr/chronologie/antiquite>

<http://serge.mehl.free.fr/>

<https://www.victorian-cinema.net/machines>

<http://toastyttech.com/guis>

<https://aosabook.org>

<http://waterbearlang.com>

<https://fr.wikipedia.org/>

https://en.wikipedia.org/wiki/Timeline_of_computing

https://en.wikipedia.org/wiki/History_of_the_graphical_user_interface

<https://www.youtube.com/watch?v=8KHuSw0W6OA>

<https://www.youtube.com/watch?v=FlfChYGv3Z4>

https://www.youtube.com/watch?v=eMy4vSZ-J_I

<https://www.youtube.com/watch?v=MQzpLLhN0fY>

<https://www.youtube.com/watch?v=YBnBAzrWeF0>

<https://www.youtube.com/watch?v=2dKG21u2aSo>

<https://www.youtube.com/watch?v=Yc945sNB0uA>

<https://www.youtube.com/watch?v=YyxGIbtMS9E>

<https://www.youtube.com/watch?v=6TRfy70DqD8>

<https://www.youtube.com/watch?v=h9H6WkUeuUY>

<https://www.youtube.com/watch?v=eIpoA7Ir9p8>

<https://www.youtube.com/watch?v=7XTHdcmjenI>

https://www.youtube.com/watch?v=Ag1AKII_2GM

https://www.youtube.com/watch?v=uL1FF_iwpr4

<https://www.youtube.com/watch?v=avTMg2THEvM>

<https://www.youtube.com/watch?v=RQYuyHNLPTQ>

Code

```
import pandas as pan;
import numpy as np;
import io;
import matplotlib.pyplot as plt;

f = io.open("HistoirePhilosophieInformation.md", mode="r", encoding="utf-8")
initialText = f.read()
formattedText = initialText;

charactersToRemove = ["#", "*", ",", ".", "]
for char in charactersToRemove:
    formattedText = formattedText.replace(char, "")
    initialText = initialText.replace(char, "")

charactersToReplaceBySpace = ["'", "\""]
for char in charactersToReplaceBySpace:
    formattedText = formattedText.replace(char, " ")

def removeEmptyStrings(item):
    return item != ""

def displaySerie(serie):
    print(len(serie.index))
    plt.figure(figsize=(20, 5))
    plt.xticks(rotation=90)
    plt.plot(serie.index, serie.values)
    plt.show()

def splitTextToWords(lines):
    return pan.Series(
        list(
            filter(removeEmptyStrings, " ".join(lines).split(" "))
        )
    )

lines = initialText.split("\n")
formattedLines = formattedText.split("\n")
wordList = splitTextToWords(lines);
formattedWordList = splitTextToWords(formattedLines);

values_count = formattedWordList.value_counts();

values_more_than_5_letters = values_count[values_count.index.str.len() > 5].loc[lambda x : x > 20]

print(f'Nombre de mots : {len(wordList)}');
displaySerie(values_more_than_5_letters)
```


Lexique

Ce mémoire est composé de plus de 46000 mots, dont ~9000 distincts.

Ci-dessous les définitions des moins communs, surtout ceux propres au jargon informatique.

Pour toute autre définition, veuillez vous référer à un dictionnaire.

Mot	Définition
CROUS	Centre régional des œuvres universitaires et scolaires
RENATER	Réseau national de télécommunications pour la technologie, l'enseignement et la recherche
VPN	Virtual Private Network
DUT	Diplôme Universitaire Technologique
CRM	Customer Relation Management, c'est-à-dire en français, gestion de la relation client
SQL	Structured Query Language, système et langage permettant de structurer et requêter une base de données.
PHP	PHP Hypertext Preprocessor, langage objet permettant de réaliser un système de gestion de l'information renvoyant de HTML
HTML	HyperText Markup Language, langage de balises structurant un document hypertexte.
CSS	Cascade Style Sheet, langage définissant l'apparence des éléments d'un document hypertexte.
PNG	Portable Network Graphics : Format plus léger que les Joint Photographic Experts Group (JPEG), spécialement conçu pour le web
SVG	Scalable Vector Graphics : Format d'image vectorielle, non définie par des pixels, mais des formes mathématiques sur lesquelles on peut zoomer à l'infini.
Complexité	https://fr.wikipedia.org/wiki/Analyse_de_la_complexit%C3%A9_des_algorithmes#Complexit%C3%A9_comparatif
LoL	League Of Legends
Cadriciel	Framework : Outil de développeur servant de base et de cadre pour la réalisation d'un logiciel
Pacte d'Ulyse	Décision prise en amont d'une situation dangereuse pour s'y préparer, en référence au chant des sirènes.

Table des illustrations

Toutes les figures ont été réalisées par mes soins.

Id	Titre	Page
F1	<u>Comparaison dpkg keyboard-configuration VS residence-manager</u>	5
F2	<u>meme sur l'expérience développeur</u>	10
F3	<u>Bâtons de Napier</u>	27
F4	<u>Définitions de la syntaxe logique</u>	35
F5	<u>Table de vérité</u>	35
F6	<u>Matrice « RACI » simplifiée, d'après mes 5 ans d'expérience dans l'industrie informatique</u>	63
F7	<u>Accès à la gestion des ports dans l'interface freebox OS 4.7</u>	73
F8	<u>Interface de configuration DNS d'Infomaniak</u>	73
F9	<u>Contenu d'un script situé dans le dossier de démarrage Windows</u>	75
F10	<u>Graphique UML discrétisé d'une voiture en tant que véhicule</u>	82
F11	<u>Comparaison d'interfaces entre celle de Swagger et ma proposition de système</u>	87
F12	<u>Comparaison VSCode vs EditIDE</u>	88
F13	<u>Prototype project-manager utilisant l'API de Gitlab</u>	89
F14	<u>Top 35 des mots de plus de 5 lettres les plus utilisés dans ce mémoire</u>	94
F16	<u>Nombre de mots par chapitre de ce mémoire</u>	95
F17	<u>Identification d'un orateur présent dans le reportage « Révoltes et révolutions technologiques » d'ARTE</u>	95
F18	<u>Diagramme de Venn pour classer une information</u>	97

1. https://www.google.fr/books/edition/Les_vies_des_hommes_illustres/iYFZAAAAcAAJ?hl=fr&gbpv=1&pg=PA142&printsec=frontcover ↵ ↵

2. \forall signifie « pour tout », quant à \wedge , voir page 35 ↵

3. (voir syllogisme page 13) ↵

4. Gary Urton, Signs of the Inka khipu : Binary coding in the Andean knotted-string records, Austin, University of Texas Press, 2003, 1re éd., 202 p. (ISBN 978-0-292-78539-7, Online Computer Library Center : [50323023](#)) ↵

5. <https://gallica.bnf.fr/ark:/12148/btv1b84470700/f95.item.zoom> ↵

6. <https://www.ancientportsantiques.com/wp-content/uploads/Documents/AUTHORS/Vitruve-Maufras%20-%20Edit%20Pancroucke%201848-Livres%20I%C3%A0V.pdf> ↵

7. <https://hal.science/hal-01609504> ↵

8. <https://archive.org/details/newtonspmathema00newtrich/page/n7/mode/2up> ↵
9. <https://archive.org/details/optickstreatise00newta/page/n9/mode/2up> ↵
10. https://archive.org/details/bub_gb_sc-Zaq8_jFIC/page/n3/mode/2up ↵
11. <https://gallica.bnf.fr/ark:/12148/bpt6k505471/f682.item> ↵
12. <https://archive.org/details/schoolsociety00deweiala/page/n8/mode/1up> ↵
13. <https://patents.google.com/patent/US359748> ↵
14. <https://patents.google.com/patent/US174465A/en?inventor=Alexander+Graham+Bell&sort=old> — <https://patents.google.com/patent/US178399A/en?inventor=Alexander+Graham+Bell&sort=old> ↵
15. Irving H. Annelis, « Peirce's Truth-functional Analysis and the Origin of the Truth Table » ↵
16. <https://courses.media.mit.edu/2004spring/mas966/Peirce%201878%20Make%20Ideas%20Clear.pdf> ↵ ↵
17. <https://worldwide.espacenet.com/patent/search/family/001739501/publication/GB593017A?q=pn%3DGB593017> ↵
18. <https://www.youtube.com/watch?v=-BP7DhHTU-I> ↵
19. <https://www.youtube.com/watch?v=7lVAFcDX4eM> ↵
20. <https://youtube.com/watch?v=zXPiqk0-zDY> ↵
21. https://store.steampowered.com/app/1444480/Turing_Complete/ ↵
22. <https://www.w3.org/History/1945/vbush/vbush.txt> ↵
23. <https://worldwide.espacenet.com/patent/search/family/009975642/publication/?q=pn%3DGB269729A> ↵
24. <https://patents.google.com/patent/US1563731A/en?inventor=Ducas+Charles> | <https://worldwide.espacenet.com/patent/search?q=pn%3DUS1563731A> ↵
25. <https://worldwide.espacenet.com/patent/search/family/009975642/publication/GB269729A?q=pn%3DGB269729A> ↵
26. <https://worldwide.espacenet.com/patent/search/family/032233881/publication/GB639178A?q=pn%3DGB639178> ↵
27. <https://worldwide.espacenet.com/patent/search/family/035202468/publication/US1745175A?q=pn%3DUS1745175> ↵
28. <https://spectrum.ieee.org/how-europe-missed-the-transistor> Michael Riordan—How Europe Missed The Transistor ↵
29. <http://wiki.c2.com/> ↵
30. https://www.youtube.com/watch?v=8DBhTXM_Br4 ↵
31. <http://rendell-attic.org/gol/utm/index.htm> ↵
32. <https://oimo.io/works/life/> ↵ ↵
33. Livre III, Chap. 2, fragment 1111a 2-20 — <https://remacle.org/bloodwolf/philosophes/Aristote/nicom2.htm#II> ↵
34. <https://www.youtube.com/watch?v=RVB3PBpXmWg> ↵
35. https://www.youtube.com/watch?v=VI_S21D7j9w ↵
36. <https://www.technologyreview.com/2023/02/14/1067869/rust-worlds-fastest-growing-programming-language/> ↵
37. <https://hal.science/hal-02970135/document> ↵
38. <https://pubs.acs.org/doi/10.1021/jacs.3c07000> ↵
39. <https://scrollprize.org/> ↵
40. <https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v2:fr> ↵
41. <https://fr.statista.com/infographie/28462/estimation-et-prevision-nombre-utilisateurs-casques-realite-virtuelle-augmentee-dans-le-monde/> ↵
42. <https://www.codingame.com/work/fr/codingame-coderpad-tech-hiring-survey-2022/> ↵
43. voir code en annexe ↵