No code et low code

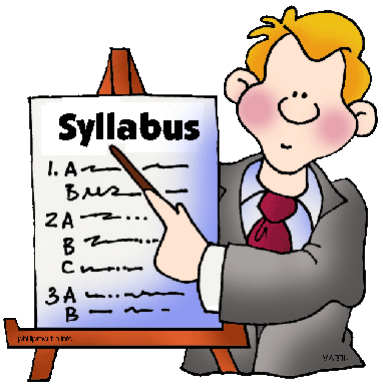
C'est sérieux ?

4 Juin 2021

En 2024, les "marketers" prévoient un chiffre d'affaires de 52 milliards$

Sommaire

No-Code et Low-Code : bonne idée ou stupidité



Le grand schisme de la programmation

- Désormais cinq familles de développeurs (on peut en imaginer d'autres)
  - Langages proches des machines : assembleurs, C…
  - Langages script et déclaratifs (SQL, HTML)
  - Langages objet et fonctionnels
  - Langages "low code"
  - Langages proches du métier (Cobol, L4G)
- Ce ne sont pas mêmes populations
- L'avenir va conforter la séparation entre des développeurs très professionnels et des développeurs occasionnels
- Quel intérêt y aura-t-il à conserver des développeurs chez soi, autres que pour des interfaces, des sites Web "jetables" ?

La programmation n'est pas une science

- La conception d'un compilateur est une science, appliquer sa syntaxe est une technique.
- Les bonnes pratiques d'un langage se réfèrent à l'expérience. On sait ou on ne sait pas.
- Le risque est grand de se réfugier derrière l'expérience d'un langage et de la confondre avec la connaissance.
- Il est beaucoup plus facile de se spécialiser sur un langage que d'essayer de prendre du recul et de le repositionner dans son contexte.
- Sujet très sensible, qui divise la communauté…
- Cela fait 40 ans qu'il y a un "gap", une méfiance, entre les développeurs et les usagers
  - Arrogance
  - Indifférence des utilisateurs par rapport là la "chose" informatique
  - Le Low code et le no code entrent dans ce contexte
- Le succès d'une méthode agile telle que Scrum tient en partie au rôle joué le PO et les équipiers…

La compétence des développeurs

- La « compétence » en matière de codage, ne se décrit plus de la même manière qu'auparavant.
- Ce qui compte, c'est d'être résilient et de répondre au plus vite aux besoins changeants des entreprises.
- La véritable compétence se situe désormais chez les prestataires, Facebook, IBM, Google, pas chez les clients, chez qui elle devient inutile.
- Quelle sera la part du « low code ?

La compétence des développeurs

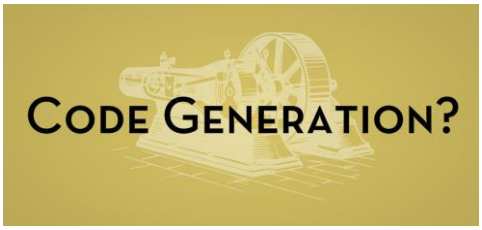
- Le développeur applique des « patterns », des recommandations, comme un joueur d'échec dans une situation donnée cherche à reproduire ce qui a déjà fonctionné.
- Pour faire communiquer des objets ou séparer l'interface utilisateur, du traitement métier et de la restitution client (MVC). Pourquoi refaire moins bien, ce qui a fait ses preuves ailleurs.
- On est aujourd'hui dans un contexte de réutilisation systématique et de généricité des composants.
- On ne se lance pas dans une application, sans s'informer des meilleures API et composants réutilisables… Imposés par l'entreprise.
  Mais… s'approprier une API et des patterns ne confère pas à celui qui le fait, le statut de scientifique de haut niveau.
- **Le codage n'est pas une science, mais en plus c'est une technique relativement simple, qui nécessite seulement d'avoir de la mémoire et de s'inspirer des bonnes pratiques déjà mises en œuvre…**
- Si on analyse le corpus d'une application, on s'aperçoit que plus de 80 % de son codage est qu'une mise en situation de composants et de patterns qui existent déjà.

Design patterns elements of reusable object-oriented software, publié en 1995, décrit 23 DP en C++ et Smalltalk (Erich Gamma, Richard Helm, John Vlissides, Ralph Johnson)

- Des problèmes récurrents qui reviennent en programmation OO, sont traités une fois pour toutes, livrés sous forme réutilisable, sans avoir à revenir sur le cheminement d'élaboration
- Un problème + sa solution = design pattern, une solution récurrente à un problème de conception
- Ce n'est pas du code
- Analogies avec le jeu d'échecs
- Les patterns font partie de la boîte à outils imposée par le chef de projet

La génération du code fait de gros progrès

- Certains observateurs, n'hésitent pas à dire que l'Intelligence Artificielle pourrait remplacer les développeurs et chefs de projets.
- C'est vrai pour la partie codage, qui n'est donc qu'une transcription d'un existant, mais inexact pour la partie conceptuelle.
- Aujourd'hui, l'IA n'est pas suffisamment mâture pour se comporter de manière autonome.
- Pour le codage, par contre, la génération progresse très vite et même si le code produit ne répond pas toujours aux canons de l'esthétisme, il faut reconnaître qu'il ne marche pas plus mal qu'un autre.

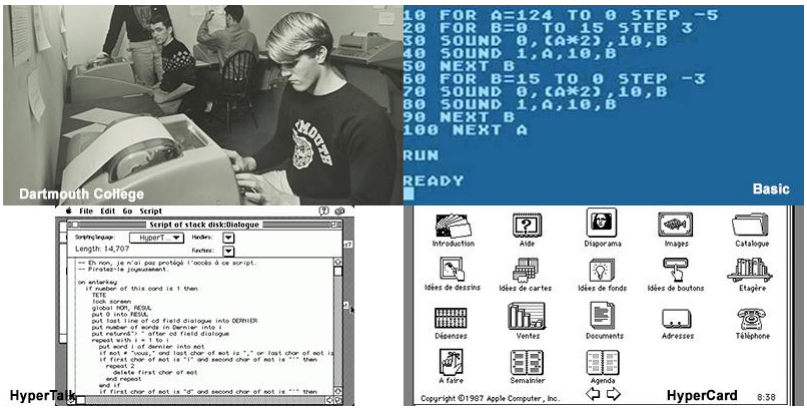


Le fantasme du code sans code

- Concept marketing qui regroupe les technologies de développement par des non informaticiens
- Microsoft et le "low code" : opération séduction
  - Il y a eu de nombreux outils qui permettaient de coder sans avoir de compétences spécifiques : Dreamweaver (Adobe), FrontPage (Microsoft)
  - Microsoft récidive sur les mobiles avec une nouvelle version de Power Apps (2015) : Power Apps FX en 2021
  - Mécanisme fondé sur des modèles, qu'il suffit de remplir, mais dont le développeur n'a pas la maîtrise de la disposition et non pas de type canevas où l'organisation des éléments est modifiable
  - Cloud Azure et connexion à Microsoft 365
- Google est présent dans le low code avec AppSheet (rachat en janvier 2020)
  - Connexions à Microsoft 365, Box, Salesforce, Dropbox, AWS DynamoDB et MySQL
- Que faut-il en penser ?
  - On n'est plus dans un contexte de risque de pollution des ressources
  - Applications jetables mobiles et BI
  - Faible efficacité, ergonomie contestable, risques de sécurité
  - Il vaut mieux s'en passer… Cela n'a jamais vraiment marché, alors…
- 50 produits et plates-formes sont déjà disponibles
- La vraie cible de ces plates-formes sont les espaces fermés tels que G Suite de Google ou Office 365 de Microsoft, pour qui elles constituent effectivement une solution intelligente de développement rapide, de ce que l'on peut considérer comme des plugins.

il y aura deux publics pour ces technologies. Le TI pour lequel on sera sceptique. Mais aussi et surtout le grand public qui va accéder au codage sans compétences particulières, pour construire un site en un quart d'heure, pour intégrer un module de paiement dans un site existant, en quelques minutes, le tout sans rien demander au TI…

Les origines du codage faiblement technique



- Origines : mai 1964, avec l'apparition du langage Basic au Dartmouth College, imaginé par John Kemeny et Thomas Kutz, pour converser avec un système de temps partagé, le DTSS ("Dartmouth Time Sharing System").
- Lancement du langage HyperTalk en 1987 par Dan Winkler, repris sur le système Hypercard du Macintosh d'Apple, par Bill Atkinson.

Quatre formes de codage non technique

Le no-code s'adresse à ceux qui veulent développer des applications, mais sans écrire la moindre ligne de code. Ce sont des plates-formes spécialisées.

Typiquement du code visuel. A partir d'un menu, on dispose sur une scène des objets qui apparaissent tels qu'ils seront en production. Le code est caché.

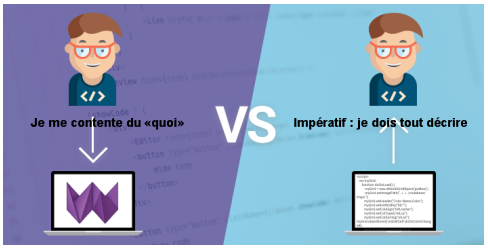
Le low-code est essentiellement une aide au développement qui vise les professionnels, pour fournir des applications plus rapidement, en allégeant leur travail, d'un certain nombre d'éléments qui deviennent transparents. Ce n'est pas le domaine le plus prometteur.

Le mode HpaPaaS est une évolution de l'aPaaS, une plate-forme de développement, proposée dans le Cloud, pour les non professionnels, mais pour des développeurs de métiers, sur laquelle sont greffées les concepts de low code : rapidité, efficacité, résilience.

- On distingue quatre formes de codage qui ne demandent pas (ou peu) de connaissances techniques : visuel, no-code, low-code et plates-formes HpaPaaS pour les entreprises.
- Trois cibles potentielles : le TI et ses développeurs professionnels, leurs utilisateurs, monsieur et madame ToutleMonde…

Les langages déclaratifs

- Avec un langage impératif, on décrit précisément la manière dont le code va se dérouler, pour aboutir à un résultat « espéré ». On dit que l'on décrit le « comment » pour aboutir au « quoi ».
- À l'inverse, un langage déclaratif comme SQL ou HTML, on ne précise que le « quoi », sans détailler la manière pour y arriver.
- En SQL on écrira par exemple :
- *SELECT * FROM CLIENTS WHERE Nom = "Smith";*
- Autrement dit on demandera d'afficher (sélectionner) tous les clients dont le nom est « Smith », sans se préoccuper des problèmes logiques, tels que le nombre de clients dans la table ou la manière de la scruter, en faisant varier un index de 1 à "clients.nombre", par exemple, en pseudo-code.
- La plupart des langages de requêtes NoSQL sont de type déclaratif : CQL de Cassandra



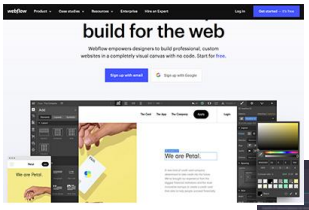
Le TI a-t-il besoin des outils low-code et no-code ?

- Faut-il mettre à la disposition des entreprises, des moyens d'écriture d'applications sans code ou avec un minimum de code, qui ne nécessiteraient pas de connaissances techniques avancées.
- Le marché dit oui : les grands éditeurs comme Salesforce, Microsoft, SAP et bien d'autres.
- Qu'en pensent les usagers ?
- On a mis la charrue avant les bœufs, pour au moins deux raisons.
  - On est d'abord dans une période de transition où 70 % des développements, en moyenne selon les pays, seront pris en charge par des indépendants, extérieurs à l'entreprise.
  - La seconde raison tient à ce que ces dispositions ne sont pas nouvelles et qu'elles n'ont jamais été véritablement exploitées.
- Il faut admettre que le codage n'intéresse pas les utilisateurs, qui ne voient dans ce "charabia" qu'une perte de temps, incompatible avec leur propre fonction.
- Ce sera un fiasco garanti...
- Ces outils ne seront pas exploités par les usagers, mais par les "gens du TI".
  - Pour concevoir un cadre applicatif, à compléter ensuite par des moyens traditionnels, pour préparer des maquettes, voire du jetable.
- Ce ne sera pas un franc succès, car les développeurs sont aussi très attachés à leurs outils et ne verront pas d'un bon œil des intrus qui leur gommeront ce qui se passe réellement derrière le rideau.
- Les développeurs aiment programmer et le no code ou low code, ce n'est pas du codage.
- Cas des PME. Les mêmes fournisseurs estiment qu'elles apprécieront de disposer de ce genre d'outils qui leur permettront de se passer des services d'un consultant.
- C'est faux. Les patrons de PME ne seront pas plus intéressés par le codage que par le TI lui-même. Ils ont besoin de solutions clés en mains, adaptées à leurs besoins et ne se transformeront pas du jour au lendemain en codeurs, même superficiels.

Le No Code

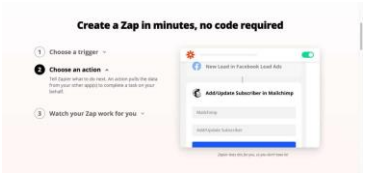
- Le **no-code** est destiné surtout aux usages hors entreprise (sauf cas particuliers)
- Il vise les allergiques au codage, qui ne veulent pas faire un minimum d'effort d'appréhension.
  - Ce sont les conducteurs de véhicules, qui ne savent pas où se trouve le moteur. Mais qui s'en accommodent fort bien.
- C'est là que se situent les vrais gisements de la technologie.
- Son marché est estimé à 4 milliards $, mais les observateurs estiment qu'il devrait monter à 21 milliards $ en 2022, voire pour d'autres, à 52 milliards $ en 2024.
- Pour un non TI, les outils semblent magiques.
  - Sans aucune connaissance, on peut fabriquer quelque chose de crédible en quelques minutes, des éléments d'interface, l'adjonction de services externes, des moyens d'accès aux données, voire l'insertion d'une fonctionnalité nouvelle dans un site Web.
  - Des outils tels que Universe ou Gumroad, sont caractéristiques, mais il en existe bien d'autres…
- En général, fonctionnent par glisser/déposer : il faut préparer les textes, images, vidéos…à l'avance et l'outil les met en scène sans nous demander de coder quoi que ce soit
- On les associe aux langages déclaratifs, au même titre qu'HTML ou que SQL.

Bubble.io



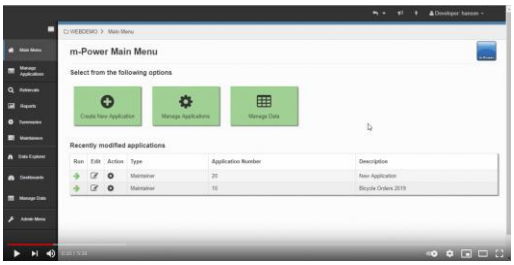
Webflow
Bravo Studio

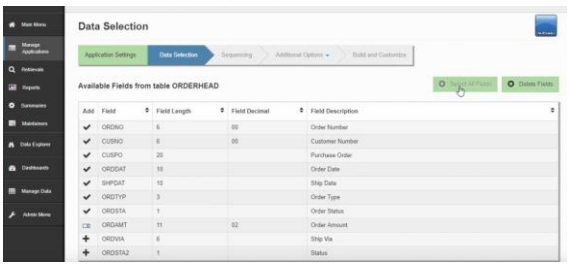
Carrd

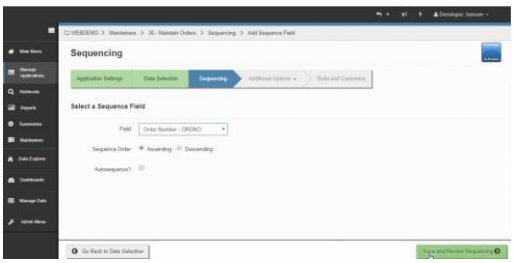


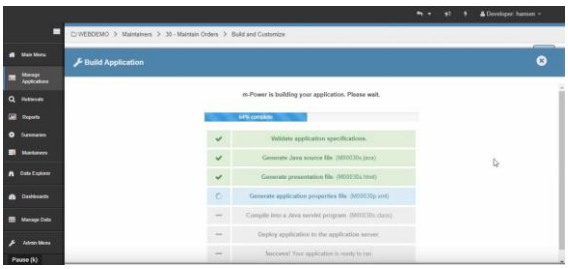
Zapier

Un exemple de No Code



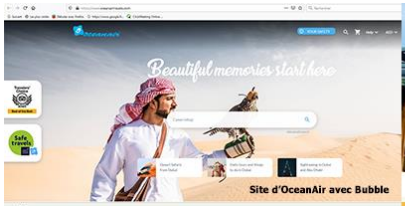






Le Low Code

- Le **low-code** est très différent du no-code : pour les mêmes applications, nécessite un minimum de connaissances techniques.
- Ce sont des solutions pour accélérer la réalisation d'une application, mais sans en gommer tous les constituants. Ce n'est plus du déclaratif.
- C'est comme si on avait mis un cache devant certains aspects du codage.
- Le mode HpaPaaS ("High Performance aPaaS"), est plus récent. Il constitue une évolution du mode PaaS, dans lequel les prestataires de Cloud fournissent toute l'infrastructure et les outils nécessaires pour effectuer un développement professionnel. A la différence de l'IaaS, les clients ne se contentent pas des services d'infrastructures, qui leur permettraient aussi de développer des applications, mais qu'ils devraient configurer comme s'ils étaient chez eux.
- Entre l'IaaS et l'HpaPaaS, il y a eu un concept intermédiaire, celui de l'aPaaS (a pour application), avec des plates-formes PaaS, mais construites pour des non professionnels, ceux qui nous intéressent ici
- En fait l'HpaPaas est une version de l'aPaaS, pour des environnements professionnels plus complets, mais rapides et conformes au low code.



Site d'OceanAir avec Bubble

Site de Walt Disney avec WordPress

Site KeyNest avec SquareSpace

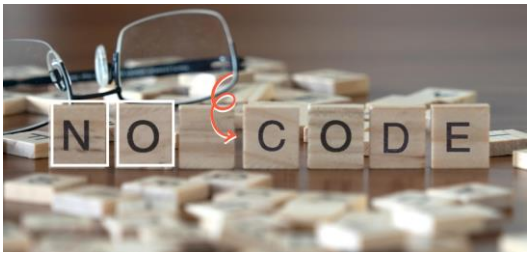
Site PrintingNewYork avec Woo

10 usages possibles

- Fonctions d'intégration et d'interfaçage, pour communiquer avec une application en place, lui proposer un formulaire et une extraction de données, s'insérer dans un processus et lui fournir un module qui n'aurait pas été prévu à l'origine.
- Développement de sites Web à faible couverture métier.
- Certains outils permettent d'intégrer des assistants. On les retrouve souvent quand il s'agit de fabriquer des mini-sites de promotion, les "landing-pages".
- Les plates-formes de service constituent un besoin récent. Elles permettent de regrouper toutes les interactions entre les fournisseurs et les clients, des fonctions de self-service, entre autres, de véritables portails, dont on peut imaginer qu'ils dépassent de loin les possibilités du low-code. Ce n'est pas ce que pensent des éditeurs comme Salesforce et Zendesk.
- Le cas des plates-formes de e-commerce. Aujourd'hui, tout le monde est susceptible de vendre quelque chose et il faut donc mettre à disposition des nouveaux venus des solutions qui englobent tout ce qu'un site de commerce doit contenir, le Web, la gestion d'un catalogue, un service de paiement avec carte bancaire, etc. Le tout sans avoir trop à s'investir dans la mécanique interne.
- Il existe quelques solutions sans doute incomplètes pour un vrai professionnel de la vente sur Internet, mais largement suffisantes dans 90 % des cas d'usage. Ex de Shopify.

10 usages possibles

- DevOps : apport d'addins sur des points précis du monitoring, de la gestion des builds, etc.
- Le domaine de l'analyse de données et de l'Intelligence Artificielle est plus contestable. Car, généralement, il concerne des populations aguerries au codage et qui n'ont pas de véritable besoin de low ou de no code.
- Ce n'est pas non plus l'avis des créateurs d'oviously.ai qui pensent qu'il y a quand même un créneau d'usage, pour des tâches plus rapides, dans la phase apprentissage surtout.
- Le cas des applications de sites "corporate" est intéressant.
  - De nombreuses grosses structures passent leur temps à s'envoyer des fichiers dans tous les sens : Excel et Word, pour constituer des "fonds de dossiers" liés à un projet ou à une décision.
  - C'est vite la pagaille et on ne sait plus qui a fait quoi et où se trouve le dernier envoi : le fait de professionnaliser tout cela, avec un entonnoir "portail" où tout est prévu et pensé, rendra de grands services. On pourra imaginer des addons de projets développés en low code.
- La fabrication des maquettes et modèles correspond à l'esprit du low code. Bien que la plupart des chefs de projets vont préférer la boîte à outils normale et ne voudront pas s'encombrer d'un utilitaire externe, les utilisateurs, voire les consultants qui auront à rédiger un cahier des charges, apprécieront l'aide du low code.
- Reste ce que le Gartner appelle les solutions aPaas et HpaPaaS, qui sont le haut de gamme du low code, destiné aux entreprises, les produits faisant partie, cette fois, de la boîte à outils, au même titre que n'importe quel compilateur.



Des solutions concrètes

Inconvénients