# DIGITALOSCOPE

Laurent Tedesco
*CEO d'Humbrain*

# Les nouvelles offres #nocode, mort ou salut des développeurs ?

**C'est une évidence, le marché du « #nocode » semble atteindre l'âge de la maturité. Fédérant les outils promettant le développement d'applications « sans écrire une seule ligne de code », et donc en se passant de spécialistes de la programmation, ce mouvement signe-t-il la fin des développeurs ? pas si sûr…**

Derrière le mot valise #nocode, les WebFlow, Bubble ou AirTable, pour ne citer qu'un petit échantillon, génèrent un véritable engouement en ce moment sur le web et les réseaux sociaux. En pratique, ce hashtag désigne une kyrielle d'outils, en mode SaaS pour la plupart, qui ont des périmètres fonctionnels et des ambitions très variés. Du simple connecteur de données, comme Sheety, à l'atelier logiciel complet comme Bubble ou le français Panda Suite, on peut trouver aujourd'hui tous les composants permettant de fabriquer, comme un jeu de Lego (ou de Mécano, pour ma génération) n'importe quelle application. Le besoin n'est pas nouveau, ces réponses ne le sont pas vraiment.

## UNE VIEILLE QUÊTE DU GRAAL

Dès l'apparition des ordinateurs civils, à l'après-guerre, leurs utilisateurs, qui étaient par essence aussi les développeurs, n'ont eu de cesse de mettre au point des interfaces ou langages, leur permettant de parler à leurs machines autrement qu'en stimuli binaires, les fameux 0 et 1. S'engage alors un long développement de langages, des Fortran ou Cobol de l'époque, aux Java, C#, Javascript, Angular ou Python d'aujourd'hui. Langages qui, bien que dits « *évolués* », restent pour le novice du charabia incompréhensible. Pour contourner ce défaut, dès les années 70, IBM propose le GAP, premier outil de l'histoire qu'on pourrait qualifier de #nocode. La méthode s'appuyait sur un formulaire standardisé, où le concepteur cochait des cases dans un agencement précis, pour exprimer son souhait de traitement. Bon, ça restait fastidieux et très éloigné de nos outils d'aujourd'hui mais c'était un début.
Dans les années 80, avec l'avènement des interfaces graphiques (Mac OS puis Windows), on voit apparaître des outils de développement adaptés à ces nouveaux univers. C'est la période de gloire des Visual Basic de Microsoft ou plus tard de WinDev en France, premiers IDE (interface de développement) à vocation #nocode. Le développeur pouvait s'appuyer largement sur les blocs graphiques préconçus et des interactivités pré-gérées et, le tout, à grand renfort de conception visuelle et de drag'n drop. Au niveau utilisateur bureautique, Microsoft Access, est déjà un tout-en-un #nocode. Avec un peu de temps, on peut en effet construire une application complète autour d'une base de données, avec des menus, des formulaires, des états, et toujours sans écrire une seule ligne de code, largement guidé par des assistants et des modèles préétablis.
L'arrivée du web, au début des années 90, a fait émerger d'autres outils #nocode : les éditeurs HTML (GoLive, FrontPage ou Dreamweaver). Ils permettaient alors de fabriquer nos premières pages web, sans écrire une seule ligne de code…HTML, seul langage reconnu par nos navigateurs.
A un niveau supérieur, les années 2000 voient se développer les CMS (Content Management System) : les Wordpress, Drupal ou Joomla. Ils permettent alors de concevoir un site web et de maintenir son contenu, là encore, sans écrire quoi que ce soit, juste en choisissant dans des catalogues d'options, de plugins ou de modèles d'interfaces (les fameux « *templates* »). On est déjà encore et toujours dans du #nocode. Ce marché a d'ailleurs donné naissance à nombre de webmasters et d'agences web, travaillant exclusivement sous Wordpress par exemple et premiers développeurs #nocode, un début de réponse à la question initiale.
L'avènement des AirTable, Bubble ou Power Apps, vrais outils qualifiés de #nocode est donc une suite logique et naturelle. Comme attendu, et même si certains d'entre eux arrivent avec de nouveaux paradigmes – comme AirTable - ils poussent simplement plus loin les pratiques de leurs prédécesseurs : interface totalement graphique, catalogues de modèles, multiples assistants et toujours la souris comme bras armé. Le Graal d'expliquer à la machine ce qu'elle doit faire sans le dire dans sa langue est-il atteint ? On en est encore loin mais on s'en rapproche.

## DES AVANTAGES ÉVIDENTS

Tout comme leurs prédécesseurs, les outils #nocode offrent la facilité, la rapidité et la souplesse dans la composition des applications. Ce qu'ils ont de plus aujourd'hui, c'est qu'ils s'inscrivent tous dans une architecture « *cloud* » avec ses paradigmes techniques et commerciaux : un hébergement externalisé et sécurisé, un déploiement de fait aisé, une montée en charge fluide et maîtrisée pour les aspects techniques et un modèle tarifaire simple puisque basé sur l'abonnement pour l'aspect commercial.

## MAIS DES DÉFAUTS ENDOGÈNES OCCULTÉS

Le premier d'entre eux est inhérent à l'environnement de ces outils : le mode SaaS offert pour la plupart soulève les problématiques de sécurité, de confidentialité et de réversibilité des données qui y sont gérées.

On a du mal à imaginer une entreprise développer une application critique pour son activité sur une plateforme dont elle ne maîtrise pas la solidité et la pérennité.

Le deuxième défaut est propre à toutes les solutions standards promettant de répondre à des besoins individualisés, à partir d'une mécanique unique, écueil vécu par tous ceux qui s'y sont déjà frottés.
C'est le fréquent piège du 80/20 ou 90/10 : L'outil va permettre de résoudre aisément et rapidement 90% de votre problème, et les 10% restant vont vous pourrir la vie, vous faire perdre temps et argent, voire, vous amener à abandonner le projet s'ils s'avèrent critiques.

Le troisième écueil, et non des moindres, est celui de la confusion entretenue dans le potentiel réel de ces outils avec les compétences d'un usager non spécialiste. Leur présentation, toujours plus séduisantes, amène à faire croire qu'on pourra tout faire avec.
Ce qui est une erreur. Car, si un bon ouvrier a toujours de bons outils, ce n'est pourtant pas l'outil qui fait l'ouvrier. Ce n'est pas parce que vous maîtriserez Word que vous deviendrez auteur de roman à succès.

Tous ceux qui travaillent – les développeurs, chefs de projet et autres PO – dans la conception d'applications le savent bien. Outre du temps dédié, leurs activités imposent des dispositions mentales particulières, d'abstraction pour la conception, de raisonnement algorithmique pour le codage organique, qui ne s'improvisent pas et/ou ne peuvent émerger simplement par ces outils.
Combien d'utilisateurs bureautiques, forcément un peu curieux et volontaires, ont cédé aux sirènes du #nocode version Access par exemple, et qui ont souffert – en temps et énergie – pour arriver à un résultat potable, quand ils l'ont atteint ?

**Alors, mort ou salut ?**

Evidemment, ni l'un ni l'autre. L'offre actuelle a tout pour séduire un large éventail de nouveaux praticiens et développer encore de nouveaux profils de développeurs.
C'est donc un marché qui s'élargit. Et puis, si les premiers chantiers ne révèlent pas de carrières de développeurs, comme on a pu le voir avec les logiciels antérieurs, ça fait toujours du travail de maintenance et d'évolution pour des spécialistes.
Et ne l'oublions pas, ceux qui ont réussi et fait fortune à l'époque de la ruée vers l'or, ce ne sont pas les chercheurs mais les vendeurs de pelles et de pioches.

**Retrouvez, commentez et partagez cet article sur digitaloscope.com**