Re: Question sur votre journée type

Bonsoir Ousmane,

Ci-dessous mes réponses aux questions que je reçois souvent, si tu as envie de plus de détails n'hésite pas à m'écrire, ou on peut s'appeler également.

Concernant le métier de développeur web spécialement, je dirais qu'il y a deux aspects: le code en lui même, et l'expérience utilisateur. Selon les personnes, on est plus ou moins bon dans ces deux domaines. Pour le premier, il faut creuser les frameworks javascript (react, vue par exemple). Pour le second, il faut penser à qui seront les utilisateurs (des jeunes, des vieux, des professionels, etc), ce qu'ils auront besoin de faire (passer le temps, acheter un objet, trouver une information, etc) et comment (rarement, souvent, en rentrant beaucoup d'information, en payant, etc). Un bon exercice c'est de regarder les sites que tu visites et de te demander comment tu offrirais le même service mais avec une interface différente.

A bientôt,

Nathan

Comment êtes-vous arrivé au poste de développeur ?

Après un Bac L et deux ans de prépa littéraire, j'ai travaillé un an dans la production de spectacle. C'est un milieu passionnant, mais je me suis rendu compte que je retournais tout le temps aux ordinateurs. J'ai donc visé de faire un Master MIAGE, une formation mi informatique mi business, ce qui me convenait parfaitement. Comme elle commençait en troisième année de licence, j'ai fait deux années d'Administration Economique et Social. Malheureusement mon dossier n'a pas été accepté en MIAGE, et j'ai donc décidé de me lancer entièrement dans la programmation en intégrant l'école EPITECH, dont je suis sorti diplomé cinq ans plus tard.

La découverte de la programmation a été un bonheur extraordinaire, c'est absolument magique :-)

Quels sont les profils recherchés par les entreprises c'est-à-dire le niveau d'études, les compétences requises et les qualités ?

Je ne me permettrais pas de parler des entreprises en général, juste de mon expérience en tant que manager et de développeur

Ce que je cherche avant tout, c'est la curiosité et la passion, car c'est un métier où il faut s'accrocher pour trouver la solution. Ensuite, je cherche des personnes avec des capacités fortes à la résolution de problème ("problem solver") et à la pensée analytique. Une fois ces deux piliers validés, je privilégie les personnes avec des profils athypiques (reconversion, parcours chaotiques, etc), mais c'est mon choix de manager de monter une équipe de touches à tout généralistes.

Par ailleurs, je note une vraie différence entre les personnes qui pensent à l'utilisateur final et celles qui sont seulement intéressées par la technique, il faut les deux profils pour faire une bonne équipe, c'est une information importante.

Le niveau d'étude après deux trois ans de formation n'a pas réellement d'importance (sauf pour décrocher un entretien je suppose), par contre les premières années font une vraie différence, même quelques mois changent la donne énormément. C'est tout le problème des formations en six mois, à mon sens elles sont trop courtes pour être efficaces, il faudrait quelques mois de plus !

Quelle formation est intéressante ?

Comme j'écrivais juste au dessus, six mois me semble un absolu minimum, et si possible choisir une formation qui aborde des vrais langages de programmation, il est inutile à l'heure actuelle de passer plusieurs semaines sur du HTML/CSS, par contre le Javascript/Ruby/Python sont des langages suffisamment souples pour ouvrir de nombreuses portes.

Quelque soit la formation, faire ses propres projets et apprentissages est vital, surtout trouver un moyen de les communiquer (Github, blog), car en tant que recruteur, c'est extrêment difficile d'évaluer les CV de

personnes débutantes; tout ce qui fait sortir du lot est à mettre en avant !

Quelles sont vos activités principales dans le développement? Comment se passe une journée type ?

Je n'ai pas de journée type car je suis à la fois manager, développeur et devops (oui c'est trop !), selon les jours j'ai plus ou moins des composantes suivantes:

- réunions pour discuter des spécifications (analyser le problème et décider de comment on va le traiter: quelle interface graphique, quel code sur les serveurs, quelles contraintes techniques, etc). Les meilleurs réunions comprennent des personnes de plusieurs spécialités: développeur application mobile, web, serveurs; des commerciaux; des personnes du support, etc.

- programmation pure : une fois qu'une fonctionnalité est analysée, il faut la coder. En général je divise en plusieurs petits morceaux que je code les uns après les autres, afin de pouvoir les tester. Il faut aussi réfléchir à comment intégrer le nouveau code avec ce qui existe déjà, et aussi se réfreiner de tout réécrire :-)

- administration système: s'assurer que les serveurs tiennent la charge, mettre en place les outils qui vont faciliter la vie des équipes (infrastructure de test automatisée par exemple, ou encore permettre aux équipes support client de faire plus d'actions), réfléchir à de nouvelles infrastructures, détecter des problèmes en amont...

- corrections en urgence de bugs en production: malheureusement il y en a toujours !

- répondre aux demandes des équipes de support client: analyse approfondie de remontée client, détails sur des fonctionnalités, etc

- répondre aux questions de mes collègues et avoir une oreille ouverte sur les discussions autour de moi pour assurer la coordination du travail de chacun, ainsi que le bien être des membres de l'équipe.

- réunions avec mes équipes (toutes les semaines, je passe au moins 30mn avec chacune des personnes que j'encadre), formation sur des nouvelles technologies et concepts.

- attendre les réponses à mes questions: je parle beaucoup à mon canard en plastique c'est très efficace de lui expliquer les problèmes, mais des fois j'ai besoin du retour d'autres personnes qui ne sont pas disponibles dans la seconde.

La majorité des actions se passent en équipe, ou sont tournées vers d'autres personnes. Quelques actions que je fais seul:

- une partie de la programmation: pour moi, programmer reste une activité souvent solitaire, il y a toujours un moment où on se retrouve seul face à l'écran, avec un carnet à côté. Programmer à deux (le "pair programming") est difficile, il faut que les deux personnes se connaissent et se complétent très bien pour que ce soit efficace.

- les relectures de code: pour relire le code de quelqu'un, il faut se plonger dans la fonctionnalité qui est attendue, relire les spécifications, puis relire le code pour bien le comprendre et l'analyser. J'ai tendance à le faire seul pour commencer, puis souvent j'invite le/la programmeur avec moi, afin d'échanger sur le code.

- recherche sur les sujets: j'adore apprendre, et je passe beaucoup de temps à approfondir ou découvrir de nouveaux sujets, technologies, il y a toujours à découvrir !

Quelles sont les aspects positifs et négatifs de votre expérience de développeur ?

En négatif, la communication est un effort permanent, en particulier envers des personnes non techniques qui ont du mal à se détacher de leur cas particulier et à penser à toutes les possibilités. Les choses sont souvent plus difficiles qu'il n'y paraissait avant, et c'est souvent frustrant.
Les bugs sont un problème quotidien, et la réalité est toujours difficile (voir impossible) à transformer en système informatique, on ne fait que la simuler de manière imcomplète.
La pile de technologies à connaître pour une nouvelle développeuse ne fait globalement qu'augmenter dès qu'on veut sortir du b.a.b.a. C'est passionnant d'avoir tout le temps de nouvelles choses à découvrir, mais aussi un peu inquiétant et écrasant.
Je travaille en start up, et nous avons souvent des dates butoirs très courtes pour réaliser certains projets, il y a donc souvent du stress. Pour le réduire, il faut travailler en amont avec les équipes commerciales, le

service client, etc. Il faut aussi améliorer la qualité de notre travail pour réduire le nombre de bugs et dégager du temps. Un excellent livre sur le sujet est "The Phoenix Project" (uniquement en anglais malheureusement).

En positif, le bonheur que je ressens à programmer, je construits des cathédrales (ainsi que des bazars) de code, c'est excitant et passionnant ! Il y a toujours à apprendre et découvrir, à recoder pour faire mieux ou différemment.
Grâce à la programmation, on peut tout faire (enfin presque), c'est rarement facile mais on peut arriver au point d'aider les utilisateurs dans leurs tâches. On se rend compte aussi qu'automatiser un humain, c'est très difficile et que nous sommes des animaux très intelligents :-) Il y a une vraie fierté dans la construction d'un système qui fonctionne bien, à tous les niveaux, que ce soit au fin fond du système ou dans l'interface utilisateur.

Quel est le salaire d'un développeur ?

voici un lien vers une étude des salaires de 2020
: https://drive.google.com/file/d/10l1_S1_LIT85onb4QR4k6M_agBryMEx2/view?usp=sharing

Des conseils pour une personne en formation ?
Coder le plus possible :-)

Faire des petits projets personnels, et les publier sur github/gitlab, le but est de montrer que vous en faites plus que la moyenne, que vous essayez des choses, ce n'est pas grave si les projets ne sont pas originaux ! Faites un clone de plusieurs sites que vous aimez bien en vous concentrant sur les fondamentaux, faites un programme qui fait le travail de vos amis, ce que vous voulez !

Pour inspiration, voici quatre projets célèbres qui ont commencé par des étudiants, qui comme vous, voulaient apprendre: Linux, Redux, Jquery, Axios

Discuter de votre code avec d'autres personnes, pour obtenir des retours et progresser. Par exemple avec des gens de votre promo pour commencer.

Quelques conseils de code à creuser sur le net:

- "make it work, make it right, make it fast": d'abord faire un proto, puis faire passer les cas spéciaux, et seulement à la fin s'intéresser aux performances

- Do Repeat Yourself: si vous tapez 2 ou 3 fois le même morceau de code, mettez le dans une fonction

- Keep It Stupid Simple et You Aren't Gonna Need It: ne pas essayer de prévoir trop loin dans le futur et ne pas faire du code compliqué avant d'en avoir vraiment besoin

- responsabilité unique: une fonction, une classe, devrait faire idéalement une seule chose. Faites des petites fonctions, faites apparaître la structure du code.

- the zen of python: http://www.thezenofpython.com/ De mon point de vue cela fonctionne pour la plupart des cas, et devrait même s'appliquer à la vie en générale ;-)

- si vous êtes bloqué sur un sujet plus d'une heure, faites une pause. Plus de 2h, allez en discuter avec quelqu'un. C'est incroyable le temps perdu sur des détails tous simples qui sont flagrants pour les autres !

Auto-entrepreneur ou travail en entreprise ?

Il y a une question de caractère, c'est bien évident ! Au-delà du statut, le plus important est d'avoir un réseau de personnes avec qui échanger, pour progresser, se débloquer.

Avez vous des conseils pour retenir facilement les différents langages de programmation ?

L'habitude et la théorie.

Le plus difficile est au début, la première découverte de nouveaux concepts: programmation impérative (C), objet (Python, Ruby, Js), fonctionnelle (Clojure), déclarative (SQL, HTML, CSS). (Pour les pointilleux, le Js utilise des prototypes).

Une fois que vous avez rencontré les concepts quelques fois, apprendre un nouveau langage sera beaucoup plus facile.

Et pour finir, découvrir régulièrement un nouveau langage/concept en dehors de ce qu'impose le travail aide beaucoup.

Attention, au début il y a tellement de choses à découvrir qu'il est facile de s'y perdre ! Assurez vos bases avant d'en ajouter une nouvelle couche (du transistor au cloud, il y a le choix !).
Quelles sont vos astuces lorsque vous faites face à un problème ?

Dans mon équipe, j'interdis de rester bloqué sur un sujet plus de deux heures sans en parler à quelqu'un d'autre, cela fonctionne bien !

De manière générale, le simple fait de réfléchir à comment on expliquerait le problème à quelqu'un permet souvent de trouver la solution. Personnellement, j'ai un canard en plastique sur mon bureau, à qui je parle souvent, il m'aide beaucoup !

Apprendre à utiliser un debugger (plutôt qu'afficher des informations dans la console), permet de naviguer dans la logique de son programme pas à pas et de mieux le comprendre, tout en voyant l'évolution des variables.

La démarche scientifique, comme souvent, est la meilleure pour debugger:

- faire une hypothèse (quand on appelle cette fonction, ça crash)

- la tester (est-ce que quand je l'appelle, ça crash ?)

- analyser les résultats (oui ça crash à la ligne 13)

- chercher à invalider le résultat (en fait, le disque dur était plein, la fonction marche bien)

=> il faut accepter la réalité et ne pas se laisser emporter par des intuitions qui ne reposent pas sur des preuves.

Écrivez, faites des dessins, des diagrammes !