7 Questions à poser en entretien de recrutement

Un entretien, c'est comme une danse à deux. Bien sûr que tu as envie d'être embauché. Mais eux aussi ont probablement envie de te séduire sinon tu ne serais pas là. En en plus même si tu es débutant, le marché est en ta faveur.

L'entretien est l'occasion de savoir si ton projet professionnel va coller à celui de l'entreprise. Toute la subtilité est de faire sentir à l'autre que tu as aussi des exigences et qu'ils sont autant en train de passer un entretien que toi.

Si je n'ai qu'un seul conseil à te donner pour réussir ton entretien :
Montre de l'intérêt.
Que ce soit pour le projet, pour l'entreprise, pour les gens, pour les technos. Mais montre de l'intérêt.
Et quel meilleur moyen de montrer de l'intérêt que de poser des questions ?
Tu ne dois pas partir d'un entretien sans avoir posé au moins 3 questions !

Les questions que tu poseras seront le reflet de tes préoccupations. Donc par pitié ne commence pas par demander quelles sont les horaires de travail, ni le nombre de RTT dans l'année. Bien sûr c'est important, mais plutôt vers la fin de l'entretien qu'au début !
Je te propose là 7 questions à poser pour t'aider à évaluer le projet dans lequel tu vas bosser.
Elles sont axées sur la technique et sont faciles à poser. Ce sont aussi des questions qui permettent d'enclencher des discussions autour de sujets classiques.

C'est parti !

1. Quel gestionnaire de source est utilisé ?

Cette question est facile pour ton interlocuteur. Elle a l'avantage de montrer que tu t'intéresses à la chaîne d'outils utilisés et que c'est important.
Dans la plupart des cas, tu auras 'git' comme réponse.
Mais tu peux avoir une réponse comme mercurial ou un autre outil récent. C'est l'occasion de demander ce qui les a poussé à faire ce choix.
L'air de rien, un gestionnaire de source dit déjà beaucoup de choses sur la culture de la boite.
Si par contre tu as une réponse comme SVN ou CVS, tu peux creuser pour comprendre ce qu'il fait qu'ils restent sur des outils obsolètes.
Et si tu as une réponse du genre : 'on utilise Dropbox pour partager le code', fuis.

2. Quelle est l'ancienneté du code du projet sur lequel tu vas bosser ?

On reste dans les questions faciles. L'âge du projet et la taille de l'équipe qui y a contribué t'aidera à évaluer la taille du code existant.
S'il est ancien et que plusieurs développeurs sont passés dessus, tu sais que tu auras probablement un code inconsistant et plus ou moins documenté.
C'est pas forcément grave, mais au moins tu sais en quoi t'en tenir.

3. Combien de bugs sont recensés à date ?

Là on commence à entrer dans les questions qui peuvent piquer un peu. C'est l'occasion de comprendre comment sont remontés les bugs, comment ils sont traités, quel outil est utilisé pour les gérer. Tu verras s'il y a une équipe de testeurs dédiés, quelle est le niveau de qualité attendu.
Tu vas avoir plusieurs types de réponses :

- Les experts du truc : ils mesurent tout et tracent tout. Ils savent te dire combien il y en a, et probablement qu'il y en a beaucoup d'ailleurs.
- Ceux qui s'en occupent pas : les bugs ne sont pas suivis. Les utilisateurs ? C'est qui ça ?
- Ceux pour qui ce n'est pas un problème : les bugs ? Il y en a une poignée. Non on ne les suit pas trop : on les corrige au fil de l'eau.

Globalement, ce qui est intéressant, c'est de comprendre si les bugs sont un problème ou pas dans cette équipe. Si c'est le cas, tu peux te poser des questions sur la qualité de ce qui est produit et ce à quoi va ressembler ton quotidien. Et paradoxalement, une équipe qui passe beaucoup d'énergie à tracer les bugs et les gérer est plutôt une équipe qui a un problème avec les bugs. Sinon, elle n'y passerait pas autant de temps.

4. Quelle bibliothèque de tests unitaires et fonctionnels est utilisée ?

Là on commence à entrer dans les questions qui piquent vraiment. La formulation de la question est importante : elle laisse penser qu'il est normal d'écrire des tests automatiques. Et c'est ce qui peut mettre mal à l'aise ton interlocuteur car il n'en utilise probablement pas.
Dans la majorité des cas, tu auras une réponse comme 'on a écrit quelques tests avec xUnit', remplacer 'x' par ton langage. C'est le signe que les tests ne font pas partie de la culture de la boite.
Est-ce un souci ? Oui et non.

- Non si tu sens qu'il y a une envie de changer sur cette question.
- Oui car tu sais que tu vas bosser dans du code legacy, c'est à dire un gros pot de code dans lequel tu vas probablement ramer et t'enliser. Cette affirmation est à pondérer par la réponse à la seconde question : plus le projet et vieux et plus tu vas ramer. S'il n'y a que quelques semaines d'ancienneté, il est peut-être encore temps de mettre en place les tests facilement.

Globalement, cette question est l'occasion d'échanger sur les tests, leur place dans le développement et ce qu'ils apportent. Il est probable que tes interlocuteurs soient intéressés par les tests, mais ne savent pas les mettre en oeuvre. Si tu sens qu'ils se posent des questions, c'est bon signe. Si par contre tu sens clairement que les tests sont perçus comme du bonus et que ça prend du temps, tu sais que ton quotidien va être compliqué.

5. Quelle est la couverture du code actuelle ?

Tu auras probablement une réponse du genre ‘je ne sais pas’ qui signifie : ‘comme on a pas de tests, on a jamais mesuré ça’. Si au contraire on te donne un chiffre, c’est déjà bon signe.
S’il est en dessous de 10%, tu sais que les tests ne sont pas encore ancrés dans cette boite. Entre 10 et 90%, tu sais qu’il y a une prise en compte intéressante. Plus de 90%, ils travaillent en TDD. Et là c’est bien parti.
C’est l’occasion de poser quelques questions sur l’intégration continue : y-en-a-t’il une ? Qui s’en occupe ?
S’il n’y a pas d’intégration continue, je chercherais à comprendre pourquoi : est-ce une question d’intérêt, de temps, de compétence ?

6. Comment se prend une décision d’architecture ?

“Si par exemple il faut ajouter une brique dans l’infra ou faire un changement important d’API, comment ça se passe ?”
Là l’enjeu est de comprendre comment se réparti le leadership et quelle sera ta place dans cette histoire. Vas-tu être un simple exécutant qui suivra les ordre d’un architecte omniscient ou vas-tu être impliqué dans ce genre de décision ?
Tu pourras évaluer quelle est le niveau d’autonomie de chaque développeur et voir s’il y a des tech lead dans l’équipe.

7. Combien de temps est alloué à la veille technologique / formation ?

Il s'agit là de comprendre comment est perçue l'auto-formation et quelle est sa place par rapport au business. Tu peux demander s'il est possible de suivre des formations, comment elles sont financées.
Si tu dois mettre en oeuvre une nouvelle technologie, quels moyens seront mis en oeuvre pour t'y aider.
Assure toi d'avoir des réponses concrètes. Pas de vagues 'oui bien sûr, on t'aidera'. C'est l'occasion de voir s'il y a une règle précise ou si c'est géré au cas par cas.
Un réponse du genre : « chaque personne peut allouer une demi-journée par semaine de son temps à la veille » sonnera différemment de « on gère les demandes au cas par cas ».

Voilà pour les questions que je voulais te proposer. Je suis impatient de ton retour : ça a donné quoi en entretien, quelle type de réponse as-tu reçu ?
Si certains points manquent de clarté et tu veux des précisions complémentaires, écris moi directement sur benoit@artisandeveloppeur.fr.

Si ce n'est pas déjà fait, pense à nous rejoindre sur https://artisandeveloppeur.fr/